МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ВАСИЛЯ СТУСА
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ І ПРИКЛАДНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КОХАН ДЕНИС ЮРІЙОВИЧ

Допускається до захисту:
в.о. завідувача кафедри
інформаційних технологій,
д-р. техн. наук, професор
_____Наталія ВЕСЕЛОВСЬКА
«____»_____ 2026 р.

АНАЛІТИЧНЕ МОДЕЛЮВАННЯ В СИСТЕМАХ БІЗНЕС-АНАЛІТИКИ З ІНТЕГРАЦІЄЮ РІЗНОРІДНИХ ДЖЕРЕЛ ДАНИХ

Спеціальність 122 Комп’ютерні науки

Кваліфікаційна (бакалаврська) робота

Керівник:
Петро НІКОЛЮК, професор кафедри
інформаційних технологій,
д-р фіз.-мат. наук, професор
_____

Оцінка: _____ / _____ / _____
(бали/за шкалою ЄКТС/за національною шкалою)

Голова ЕК: _____
(підпис)

Вінниця – 2026

АНОТАЦІЯ

Кохан Д.Ю. Аналітичне моделювання в системах бізнес-аналітики з інтеграцією різнорідних джерел даних. Спеціальність 122 «Комп'ютерні науки», освітня програма «Комп'ютерні науки». Донецький національний університет імені Василя Стуса, Вінниця 2026.

У кваліфікаційній (бакалаврській) роботі досліджено теоретичні засади бізнес-аналітики та методи інтеграції різнорідних джерел даних. Проаналізовано архітектурні підходи до організації аналітичних сховищ і обґрунтовано вибір технологічного стеку. Результатом роботи є повноцінний аналітичний пайплайн для e-commerce платформи, що охоплює ELT-процес на основі Apache Airflow, хмарне сховище Google BigQuery, трансформаційний шар dbt та інтерактивний дашборд Power BI з 21 DAX-мірою і п'ятьма аналітичними сторінками. Розроблена система забезпечує інтеграцію чотирьох різнорідних джерел даних і підтримує повний цикл аналізу бізнес-показників.

Ключові слова: бізнес-аналітика, ETL/ELT, хмарне сховище даних, Apache Airflow, Google BigQuery, dbt, Power BI, схема «зірка».

66 ст., 16 рис., 4 табл., 9 дод., 45 джерел.

ABSTRACT

Kokhan D.Y. Analytical Modeling in Business Intelligence Systems with Integration of Heterogeneous Data Sources. Specialty 122 «Computer Science», educational program «Computer Science». Vasyl Stus Donetsk National University, Vinnytsia, 2026.

In the qualification (bachelor's) work, the theoretical foundations of business intelligence and methods of heterogeneous data source integration are investigated. Architectural approaches to analytical data warehouse organization are analyzed and the technology stack selection is justified. The result of the work is a complete analytical pipeline for an e-commerce platform, encompassing an ELT process based

on Apache Airflow, Google BigQuery cloud data warehouse, a dbt transformation layer, and an interactive Power BI dashboard with 21 DAX measures and five analytical pages. The developed system integrates four heterogeneous data sources and supports the full cycle of business metrics analysis.

Keywords: business intelligence, ETL/ELT, cloud data warehouse, Apache Airflow, Google BigQuery, dbt, Power BI, star schema.

66 p., 16 fig., 4 tabl., 9 append., 45 sources.

ЗМІСТ

ВСТУП

Актуальність теми дослідження. Стрімкий розвиток цифрової економіки та зростання обсягів даних ставлять перед сучасними підприємствами принципово нові вимоги до систем аналітики та підтримки прийняття рішень. За оцінками Dimension Market Research, опублікованими у листопаді 2024 року, обсяг світового ринку Business Intelligence у 2024 році становив близько 33,9 млрд дол. США і, за прогнозами, досягне 75,7 млрд дол. США до 2033 року при середньорічному темпі зростання 9,3%. Таке прискорення зумовлене насамперед масовим переходом організацій на хмарні аналітичні платформи, зростаючим попитом на аналіз даних у режимі реального часу та інтеграцією штучного інтелекту у BI-інструменти.

Водночас більшість підприємств зберігають дані у кількох несумісних системах: реляційних базах даних, файлових вивантаженнях CRM-систем, журналах подій веб-аналітики та таблицях. Інтеграція таких різнорідних джерел є однією з ключових проблем сучасної аналітичної інженерії. Огляд досліджень у цій галузі, опублікований у 2024 році в журналі Data in Brief, підтверджує, що гетерогенність, неоднорідність форматів та потреба в автоматизованому оновленні залишаються важливими викликами для побудови надійних аналітичних пайплайнів.

Поширення хмарно-орієнтованих інструментів – зокрема Apache Airflow як платформи оркестрації, Google BigQuery як колонкового сховища даних та dbt (data build tool) як засобу трансформації даних – сформувало новий стандарт побудови аналітичних платформ, відомий як Modern Data Stack. Цей підхід дозволяє реалізувати повний аналітичний пайплайн з чітким розмежуванням відповідальності між компонентами, підтримкою контролю якості даних на кожному етапі та можливістю повторного виконання будь-якого кроку обробки. Проте попри широке галузеве застосування, академічні роботи, що комплексно охоплюють проєктування та реалізацію таких систем від операційної бази до

інтерактивного дашборду на конкретній предметній галузі, залишаються нечисленними.

Викладене вище обумовлює актуальність розробки аналітичної платформи для e-commerce, яка демонструє повний цикл інтеграції різнорідних джерел даних і побудови бізнес-метрик на основі стеку Apache Airflow – Google BigQuery – dbt – Power BI.

Мета дослідження – розробити аналітичну платформу для e-commerce, яка охоплює повний цикл роботи з даними: від збору з різнорідних джерел до готових аналітичних показників в інтерактивному дашборді, та продемонструвати методологію побудови трирівневої аналітичної моделі даних засобами сучасного технологічного стеку.

Завдання дослідження. Для досягнення поставленої мети визначено такі завдання: дослідити теоретичні основи аналітичного моделювання та концепцію інтеграції різнорідних джерел даних у бізнес-системах; спроєктувати аналітичну архітектуру та операційну базу даних e-commerce платформи у третій нормальній формі та підготувати три зовнішніх джерела різних форматів (CSV, JSON, XLSX) як сукупність різнорідних джерел для ELT-пайплайну; реалізувати ELT-пайплайн на основі Apache Airflow для автоматизованого завантаження даних з чотирьох різнорідних джерел у Google BigQuery; розробити трирівневу аналітичну модель (Raw – Staging – Mart) засобами dbt та реалізувати схему «зірка»; реалізувати розрахунок бізнес-метрик засобами DAX та побудувати інтерактивний аналітичний дашборд у Power BI.

Об'єкт дослідження – процеси аналітичного моделювання в системах бізнес-аналітики з інтеграцією різнорідних джерел даних.

Предмет дослідження – методи та інструменти побудови аналітичного пайплайну на основі стеку Apache Airflow – Google BigQuery – dbt – Power BI.

Теоретичне та практичне значення одержаних результатів. У теоретичному аспекті систематизовано підходи до побудови трирівневих аналітичних моделей даних та обґрунтовано вибір хмарно-орієнтованого технологічного стеку для вирішення задачі інтеграції різнорідних джерел. У

практичному аспекті реалізовано повний аналітичний пайплайн для умовної e-commerce платформи: спроєктовано операційну базу даних PostgreSQL (27 таблиць, понад 65 000 записів), розроблено чотири DAG-пайплайни Apache Airflow для завантаження даних з різнорідних джерел у Google BigQuery, побудовано 38 dbt-моделей і схему «зірка», а також інтерактивний дашборд Power BI з 21 DAX-мірою на п'яти аналітичних сторінках. Одержані результати можуть бути використані як методологічна основа для побудови аналітичних платформ на підприємствах електронної комерції.

Апробація результатів дослідження. Основні положення і результати роботи апробовано у двох наукових публікаціях. Тези доповіді «Реалізація ETL-пайплайну для інтеграції різнорідних джерел даних у системах бізнес-аналітики» представлено на VII Всеукраїнській науково-практичній конференції здобувачів вищої освіти та молодих вчених «Прикладні інформаційні технології» (м. Вінниця, 15 травня 2026 року). Наукова стаття «Збір та стандартизація гетерогенних даних у хмарних аналітичних середовищах» прийнята до публікації у Віснику студентського наукового товариства ДонНУ імені Василя Стуса (Том 1 № 18, 2026).

Структура кваліфікаційної роботи. Робота складається зі вступу, трьох розділів основної частини, висновків, списку використаних джерел та додатків. У першому розділі розглянуто теоретичні основи аналітичного моделювання, концепцію інтеграції різнорідних джерел даних та виконано огляд сучасного технологічного стеку. Другий розділ присвячено проєктуванню аналітичної архітектури, операційної бази даних e-commerce платформи та трирівневої моделі даних. У третьому розділі описано технічну реалізацію ETL-пайплайну, dbt-трансформацій та аналітичного дашборду в Power BI, а також наведено інтерпретацію отриманих бізнес-показників. Список використаних джерел містить 45 найменувань. Додатки включають лістинги ключових фрагментів програмного коду.

РОЗДІЛ 1
ТЕОРЕТИЧНІ ОСНОВИ АНАЛІТИЧНОГО МОДЕЛЮВАННЯ ТА ІНТЕГРАЦІЇ ДАНИХ У БІЗНЕС-СИСТЕМАХ

1.1 Роль Business Intelligence у сучасному управлінні підприємством та концепція інтеграції різнорідних джерел даних

Сучасне управління підприємством нерозривно пов'язане з можливістю своєчасно отримувати, опрацьовувати та інтерпретувати дані для підтримки прийняття рішень на всіх рівнях організаційної ієрархії. Саме цю функцію виконує Business Intelligence (BI) – сукупність технологій, процесів та методологій, що перетворюють сирі дані на структуровану аналітичну інформацію, придатну для стратегічного й операційного управління [1].

Поняття «business intelligence» має тривалу історію. Вперше цей термін зустрічається у праці Річарда Девенса «Cyclopædia of Commercial and Business Anecdotes» (1865), де автор описав практику банкіра сера Генрі Ферніса, який систематично збирав комерційну інформацію і завдяки цьому приймав рішення швидше за конкурентів [2]. У науковий обіг термін ввів дослідник IBM Ганс Петер Лун у статті «A Business Intelligence System» (1958), означивши BI як здатність сприймати взаємозв'язки між поданими фактами і діяти відповідно до них [2]. Концептуальне поширення терміна в бізнес-середовищі пов'язане з 1989 роком, коли аналітик Gartner Group Говард Дреснер запропонував розглядати його як узагальнену назву методів і концепцій підвищення якості ділових рішень на основі фактичних даних [2].

Технологічна база BI формувалася поетапно. У 1990-х роках поширилися системи управління базами даних і перші ETL-інструменти, що дозволило централізовано зберігати операційні дані у сховищах даних. У 2000-х корпоративні BI-платформи компаній IBM, SAP, Oracle та Microsoft зробили аналітику доступною для широкого кола керівників. Із середини 2010-х BI-інструменти – зокрема Microsoft Power BI, Tableau та Qlik – демократизували

доступ до даних, надавши нетехнічним фахівцям змогу самостійно будувати звіти та дашборди [2]. Нинішній етап характеризується масовим переходом аналітичних систем у хмарне середовище та інтеграцією штучного інтелекту: автоматизованою підготовкою даних, аналітикою природною мовою та вбудованими прогностичними можливостями.

Відповідно до визначення Forrester Research, Business Intelligence – це «сукупність методологій, процесів, архітектур та технологій, що перетворюють сирі дані на значущу і корисну інформацію для підтримки ефективнішого стратегічного, тактичного та операційного прийняття рішень» [1]. Це визначення підкреслює системний характер BI: йдеться не лише про програмні засоби, а про цілісну методологічну основу роботи з даними.

Роль BI у сучасному управлінні підприємством реалізується через кілька функцій. Моніторинг ключових показників діяльності (KPI) забезпечує керівникам усіх рівнів єдиний погляд на результативність бізнесу в режимі, наближеному до реального часу. Аналіз тенденцій дозволяє виявляти закономірності у великих масивах даних. Сегментація клієнтської бази та товарного портфелю дає змогу приймати обґрунтовані маркетингові й асортиментні рішення. Ефективно побудована BI-система замінює інтуїтивне управління доказовим підходом, що суттєво знижує ризик помилкових рішень у динамічному ринковому середовищі [3]. Дослідження у галузі аналітики e-commerce підтверджують, що застосування BI-інструментів підвищує ефективність обробки даних і глибину аналізу бізнес-показників, що безпосередньо впливає на якість управлінських рішень [5].

Важливим елементом сучасних BI-систем є інтерактивний дашборд – візуальне середовище, що консолідує метрики з різних джерел і підтримує динамічне фільтрування та деталізацію даних. Дашборди реалізують принцип єдиного вікна для аналізу: замість роботи з розрізненими звітами керівник або аналітик отримує єдину сторінку з взаємопов'язаними візуалізаціями, де будь-яка взаємодія з одним елементом автоматично оновлює решту [6]. Це дозволяє

аналізувати дані в контексті, переходити від узагальненого рівня до деталей і виявляти залежності, які залишалися б непомітними у статичній звітності.

Для підприємств сфери електронної комерції BI набуває особливого значення. Транзакційний характер бізнесу генерує значні обсяги структурованих даних: замовлення, позиції, платежі, відвантаження, відгуки, події веб-аналітики. Паралельно існують зовнішні джерела – CRM-вивантаження з сегментацією клієнтів, прайс-листи постачальників. Без інтегрованої BI-системи ці дані залишаються розрізненими і не дають цілісного уявлення про ефективність бізнесу. Саме тому побудова повноцінного аналітичного пайплайну, що консолідує різнорідні джерела в єдину модель даних, є практичною необхідністю для сучасної e-commerce платформи [1, 4].

Сучасні підприємства генерують і накопичують дані у найрізноманітніших форматах і системах. Реляційні бази даних зберігають транзакційні записи з чіткою схемою; CRM-системи вивантажують дані у пласких CSV-файлах; веб-застосунки фіксують події у форматі JSON; постачальники надають прайс-листи у вигляді XLSX-таблиць. Кожне з цих джерел відображає лише частину реального стану бізнесу, тому отримати цілісну аналітичну картину можна лише шляхом їх інтеграції.

Різнорідні (гетерогенні) джерела даних – це сукупність джерел, що відрізняються за форматом представлення, структурою схем, семантикою полів та технологічним стеком зберігання [2]. За ступенем структурованості прийнято виділяти три категорії. Структуровані дані мають фіксовану схему і зберігаються у реляційних базах даних, CSV- та XLSX-файлах – вони добре піддаються SQL-запитам і агрегації. Напівструктуровані дані, зокрема JSON і XML, мають ієрархічну організацію з гнучкою схемою, що дозволяє представляти складні вкладені об'єкти, але потребує попередньої нормалізації перед завантаженням у реляційне сховище. Неструктуровані дані – тексти відгуків, зображення, потоки IoT-пристроїв – не мають заздалегідь визначеного формату і для аналітичного використання потребують спеціалізованих методів попередньої обробки [7]. Зазначена класифікація ілюструється рисунком 1.1.

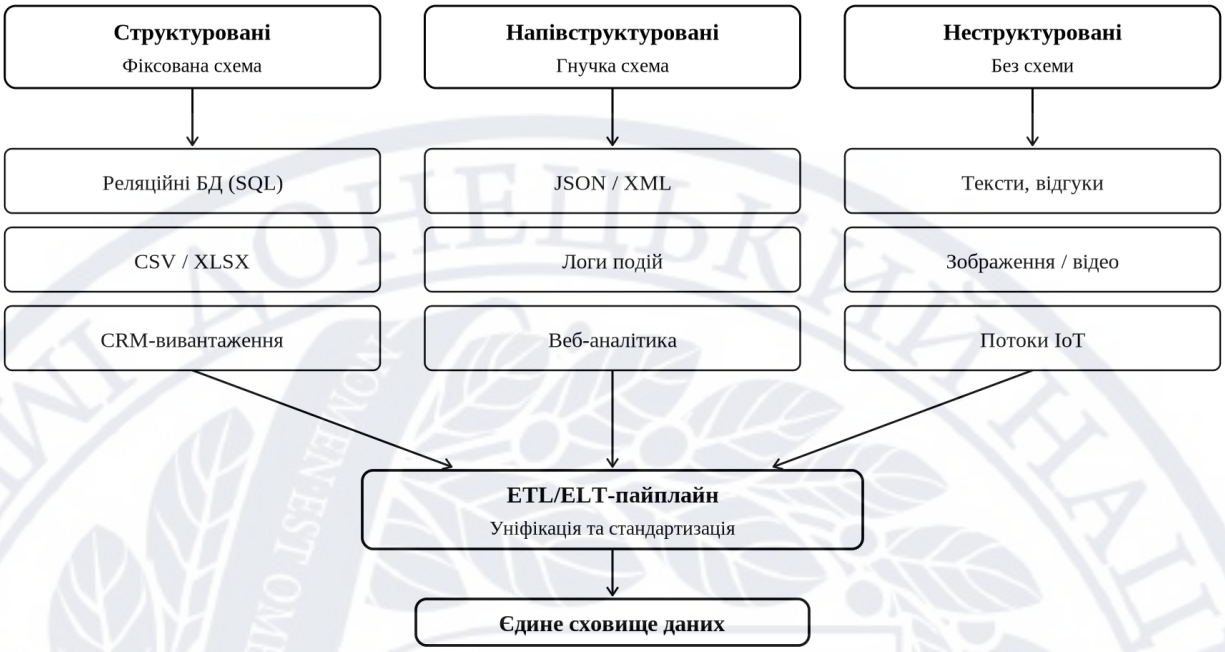

Рисунок 1.1 – Класифікація джерел даних за ступенем структурованості

Інтеграція різнорідних джерел передбачає вирішення кількох класів проблем. Структурна гетерогенність виникає, коли одна і та сама сутність по-різному моделюється у різних системах: ідентифікатор клієнта може бути цілим числом у реляційній базі, рядком у CSV-файлі та значенням NULL у JSON-події анонімного відвідувача. Семантична гетерогенність полягає у розбіжності значень однаково названих полів: поле «статус» у операційній базі може містити технічні коди («new», «paid», «delivered»), тоді як аналітична система очікує локалізованих найменувань. Технічна гетерогенність пов'язана з відмінностями у форматах дат, кодуваннях рядків та точності числових полів [8, 9]. Огляд досліджень у галузі інтеграції гетерогенних даних, проведений у 2024 році, підтверджує, що саме ці три класи проблем є найбільш поширеними при об'єднанні різнотипних джерел [2].

Для вирішення зазначених проблем у практиці аналітичної інженерії застосовується підхід ETL (Extract, Transform, Load) – послідовний процес вилучення даних з джерел, їх трансформації до єдиного стандарту і завантаження у цільове сховище. Сучасна варіація – ELT (Extract, Load, Transform) – передбачає спочатку завантаження сирих даних у хмарне сховище і лише потім

їх трансформацію безпосередньо у сховищі. Такий порядок операцій дозволяє зберегти початкові дані незмінними і повторити будь-який крок трансформації у разі помилки або зміни бізнес-логіки [1]. Саме підхід ELT покладено в основу аналітичної архітектури цього дослідження.

Ключовою концепцією при інтеграції різнорідних даних є трирівнева архітектура шарів: Raw, Staging і Mart. Шар Raw зберігає дані у точно такому вигляді, як вони надійшли з джерела. Це забезпечує можливість аудиту та повторного опрацювання без повторного звернення до джерела. Шар Staging виконує нормалізацію типів даних, уніфікацію іменувань полів і усунення технічних артефактів, характерних для конкретного джерела: приведення типів дат, відновлення символів, некоректно переданих у процесі вивантаження, видалення дублікатів. Шар Mart формує аналітичні таблиці, готові для підключення до інструментів бізнес-аналітики, із заздалегідь визначеними зв'язками між сутностями та розрахованими показниками [1].

Важливою складовою інтеграційного процесу є забезпечення якості вхідних даних. Проблеми неповноти, дублювання та невідповідності форматів є найбільш поширеними при об'єднанні різнотипних джерел і можуть суттєво спотворити результати аналітики, якщо їх не виявити на ранніх етапах пайплайну [7]. Тому шар Staging у сучасних рішеннях виконує не лише технічну нормалізацію, а й фактичну перевірку якості: виявлення пропусків у ключових полях, перевірку цілісності зв’язків між джерелами та контроль допустимих діапазонів числових значень.

У контексті цього дослідження об'єктом інтеграції є чотири різнорідних джерела, що відображають типовий набір даних e-commerce підприємства. Кожне джерело містить атрибути, яких немає в інших, тому лише їх спільний аналіз дає повну картину бізнесу. Характеристику кожного джерела наведено у таблиці 1.1.

Таблиця 1.1. Характеристика джерел даних аналітичного пайплайну

Джерело	Формат	Обсяг	Унікальні атрибути
PostgreSQL	SQL (реляційна БД)	27 таблиць, ~65 000 записів	Транзакції, замовлення, платежі, відвантаження, відгуки, персонал
customer_segments.csv	CSV	2 000 рядків	Сегмент клієнта (VIP / Regular / At-risk / New), довічна цінність (LTV), канал залучення
web_events.json	JSON	50 000 подій	Тип події, сесія, URL, пристрій, мітка часу (30% анонімних)
supplier_prices.xlsx	XLSX	154 рядки	Собівартість товару за SKU, постачальник, валюта

PostgreSQL зберігає транзакційну деталізацію замовлень, але не містить даних про сегментаційний профіль клієнта. CSV-файл доповнює клієнтський профіль метрикою довічної цінності клієнта та маркетинговим каналом залучення, що дозволяє виявляти залежність між способом первинного залучення та подальшою цінністю клієнта. JSON-файл розкриває поведінку користувача на сайті – послідовність подій від перегляду сторінки до оформлення замовлення – аж до рівня пристрою і сесії, що є основою для розрахунку конверсійних метрик. XLSX-файл надає собівартість кожного товару за SKU, що є необхідною умовою для розрахунку маржинальності. Відсутність будь-якого з цих джерел унеможливлює побудову повноцінної аналітичної

моделі: операційна база без CRM-даних не дає змоги сегментувати клієнтів, а без XLSX-прайсу неможливо розрахувати валовий прибуток.

Зазначена багатоджерельність ставить конкретні технічні вимоги до інфраструктури збору даних. Пайплайн повинен підтримувати чотири різні способи підключення до джерел: прямий SQL-запит до PostgreSQL через psycopg2, зчитування файлу CSV через pandas, розбір JSON-файлу та читання XLSX через openpyxl. Кожен з цих способів реалізується окремим DAG-пайплайном у Apache Airflow, що забезпечує незалежне керування та моніторинг завантаження кожного джерела.

Таким чином, Business Intelligence є невід'ємним інструментом сучасного управління підприємством, що еволюціонував від простих звітних систем до комплексних хмарних аналітичних платформ. Ефективна реалізація його можливостей в умовах реального бізнесу потребує вирішення задачі інтеграції різнорідних джерел даних – подолання структурної, семантичної та технічної гетерогенності засобами ETL/ELT-пайплайнів і трирівневої архітектури. Вибір архітектурних підходів і конкретних технологічних рішень для реалізації цього завдання розглядається у наступних підрозділах.

1.2 Порівняльний аналіз архітектурних підходів: OLAP-системи та хмарні сховища даних

Вибір архітектурного підходу до організації аналітичного сховища є одним із основних рішень при проєктуванні BI-системи. Важливо одразу розмежувати два поняття, які часто ототожнюють: OLAP як аналітичну парадигму та конкретний клас систем на основі багатовимірних кубів. OLAP (Online Analytical Processing) – це загальна концепція аналітичної обробки даних, орієнтована на агрегацію показників у розрізі різних вимірів. Цю концепцію реалізують як класичні системи на основі попередньо розрахованих кубів, так і сучасні хмарні аналітичні сховища (Cloud Data Warehouse, CDW) – останні виконують ті самі аналітичні навантаження, але принципово іншими технічними засобами [10].

Саме порівнянню цих двох підходів до реалізації OLAP-навантажень присвячений цей підрозділ.

Концепцію OLAP запропонував Едгар Кодд у 1993 році, виокремивши 12 правил, яким має відповідати багатовимірна аналітична система. Центральним поняттям класичних OLAP-реалізацій є куб даних – попередньо агрегована багатовимірна структура, де кожен вимір відповідає аналітичній перспективі (час, географія, товар, канал продажу), а на перетині вимірів розташовані числові показники – факти [11]. Попередня агрегація забезпечує швидке виконання стандартних звітів, оскільки система повертає вже обчислені значення замість того, щоб агрегувати мільйони рядків у реальному часі.

Проте класичні OLAP-системи на основі кубів мають суттєві структурні обмеження. Зміна аналітичної схеми – додавання нового виміру або зміна рівня деталізації – потребує перебудови кубів і повторного навантаження даних, що є трудомісткою операцією. Масштабування обмежене апаратними можливостями фізичного вузла. Підтримка різнорідних джерел вимагає окремої ETL-інфраструктури [10]. Класичні OLAP-системи добре справляються з фіксованою схемою звітності і великою кількістю одночасних запитів, проте погано пристосовані до змін – саме тому для задачі інтеграції різнорідних джерел з динамічною схемою вони не є оптимальним вибором.

Незалежно від того, яким технічним засобом реалізується OLAP-навантаження – класичними кубами чи колонковим сховищем, – постає фундаментальне питання логічного проєктування моделі даних: як організувати таблиці, щоб аналітичні запити виконувалися ефективно, а користувач міг інтуїтивно їх будувати. Відповідь на це питання дають дві методології, що заклали методологічну основу для проєктування аналітичних моделей, – підходи Білла Інмона та Ральфа Кімбала. Підхід Інмона (1990) передбачає побудову централізованого нормалізованого корпоративного сховища, з якого формуються тематичні вітрини даних. Підхід Кімбала (1996) ґрунтується на безпосередньому проєктуванні денормалізованих вітрин за принципом схеми «зірка», де центральна таблиця фактів оточена таблицями вимірів [12].

Детальний виклад методології міститься у праці Кімбала «The Data Warehouse Toolkit», яка є стандартним довідником для проєктування аналітичних моделей [12]. Схема «зірка» є найпоширенішою моделлю завдяки простоті написання запитів і ефективності агрегацій. Обидва підходи однаково використовуються як у класичних OLAP-системах, так і у хмарних сховищах.

Хмарні аналітичні сховища реалізують OLAP-навантаження принципово іншим способом. Замість попередньої агрегації у кубах вони застосовують колонково-орієнтоване зберігання: значення кожного стовпця зберігаються окремо, що дозволяє аналітичному запиту читати лише ті стовпці, які реально задіяні, суттєво скорочуючи обсяг операцій вводу-виводу [15]. Для типових аналітичних запитів, що агрегують два-три показники по мільйонах рядків, такий підхід забезпечує порівнянну або кращу продуктивність без жодної попередньої підготовки кубів.

Другою характеристикою хмарних сховищ є розмежування обчислень і зберігання. У класичних архітектурах обсяг обчислювальних ресурсів жорстко прив'язаний до обсягу зберігання. Хмарні платформи розривають цей зв'язок: зберігання й обчислення масштабуються незалежно, а вартість визначається фактичним споживанням [15]. Це дозволяє зберігати великі обсяги сирих даних за мінімальну вартість і залучати потужні обчислювальні ресурси лише під час виконання запитів.

Google BigQuery, обрана як цільова платформа у цьому дослідженні, є повністю керованим serverless-сховищем даних на базі Google Cloud Platform. На відміну від традиційних баз даних, BigQuery не потребує налаштування серверів чи управління інфраструктурою – користувач просто виконує SQL-запити, а платформа самостійно виділяє необхідні обчислювальні ресурси і звільняє їх після завершення запиту [15]. Завдяки розподіленій архітектурі та колонковому зберіганню даних BigQuery здатний обробляти запити над великими обсягами даних за секунди, що робить його природним середовищем для аналітичних трансформацій і зберігання всіх шарів пайплайну.

Serverless-модель BigQuery означає, що команда аналітиків повністю звільнена від адміністрування кластера. Оплата здійснюється за обсягом даних, відсканованих під час виконання запитів, або за зарезервованими обчислювальними слотами [16]. Для проєктів з відносно невеликим обсягом даних, як у випадку цього дослідження, безоплатний рівень BigQuery повністю покриває операційні потреби.

Порівняння класичних OLAP-систем і хмарних аналітичних сховищ наведено у додатку К.

Класичні OLAP-системи – зокрема Microsoft SQL Server Analysis Services, Oracle Essbase і IBM Cognos TM1 – ґрунтуються на кубічному зберіганні даних. Їх особливість полягає у попередній агрегації даних у багатовимірні куби: це забезпечує субсекундні відповіді при фіксованій схемі звітності і великій кількості одночасних запитів. Однак платою за таку продуктивність є висока вартість інфраструктури (власне обладнання, ліцензії, команда DBA), негнучкість схеми (зміна структури куба потребує повного перезавантаження) і обмежена підтримка джерел – переважно структурованих реляційних [10].

Хмарні аналітичні сховища (Cloud Data Warehouse, CDW) реалізують принципово інший підхід. Колонково-орієнтоване зберігання дозволяє читати лише ті стовпці, що беруть участь у запиті, знижуючи обсяг сканованих даних на порядки. Обчислення і зберігання масштабуються незалежно – платформа виділяє ресурси на час виконання запиту і звільняє їх після завершення. Трансформація даних виконується засобами SQL безпосередньо у сховищі (підхід ELT), що усуває потребу у виділених ETL-серверах. Нативна підтримка структурованих, напівструктурованих і зовнішніх джерел робить CDW природним вибором для систем з різнорідними джерелами даних [10].

Хмарні аналітичні сховища за більшістю критеріїв забезпечують суттєві переваги для сценаріїв з динамічними вимогами до схеми і різнорідними джерелами. Класичні OLAP-системи зберігають перевагу у сценаріях з фіксованою схемою звітності і необхідністю субсекундних відповідей для

великої кількості одночасних користувачів завдяки попередній агрегації у кубах [10].

Вибір BigQuery для цього дослідження обумовлений кількома чинниками. Різнорідність джерел – реляційна база, CSV, JSON та XLSX – потребує платформи з нативною підтримкою різних форматів завантаження. Трирівнева архітектура Raw–Staging–Mart передбачає виконання всіх трансформацій засобами SQL безпосередньо у сховищі, що відповідає парадигмі ELT. Нарешті, serverless-модель усуває необхідність адміністрування інфраструктури, дозволяючи зосередитися на проєктуванні аналітичних моделей.

Незалежно від обраної платформи зберігання, аналітична цінність системи визначається якістю моделі даних. Підхід Кімбала до побудови схеми «зірка» залишається актуальним і в контексті хмарних сховищ: таблиці фактів і виміри є оптимальним способом організації даних для BI-інструментів незалежно від того, де вони зберігаються [12]. Саме ця модель реалізована у mart-шарі аналітичного пайплайну цього дослідження і детально описана у розділі 2.

Таким чином, порівняльний аналіз підтверджує доцільність використання Google BigQuery як цільової платформи для аналітичного пайплайну e-commerce системи. Колонково-орієнтоване зберігання, розмежування обчислень і даних, serverless-модель та нативна підтримка SQL-трансформацій роблять його оптимальним вибором для реалізації трирівневої архітектури Raw–Staging–Mart з підключенням різнорідних джерел через Apache Airflow. Серед альтернативних хмарних сховищ Snowflake вирізняється хмаронезалежністю і підтримкою розгортання на AWS, Azure або GCP, тоді як Amazon Redshift є оптимальним для організацій, що вже використовують екосистему AWS [13]. Azure Synapse Analytics інтегрується з Power BI та іншими продуктами Microsoft, що робить його природним вибором для Microsoft-орієнтованих середовищ [14].

1.3 Огляд сучасних інструментів аналітичного моделювання: Apache Airflow, Google BigQuery, dbt та Power BI

Побудова повноцінного аналітичного пайплайну від різнорідних джерел до інтерактивного дашборду потребує координованого використання кількох спеціалізованих інструментів, кожен з яких відповідає за окремий етап обробки даних. У цьому підрозділі розглянуто чотири ключові інструменти, що складають технологічний стек дослідження: Apache Airflow, Google BigQuery, dbt та Power BI – і обґрунтовано їх вибір порівняно з альтернативами.

Apache Airflow – платформа оркестрації робочих процесів з відкритим вихідним кодом, розроблена в Airbnb у 2014 році і передана під управління Apache Software Foundation [17]. Вона дозволяє програмно визначати, планувати та відстежувати послідовності задач у вигляді спрямованих ациклічних графів (DAG – Directed Acyclic Graph). Кожен DAG описується Python-скриптом і містить набір задач з явно визначеними залежностями між ними: задача виконується лише тоді, коли всі задачі, від яких вона залежить, завершилися успішно [17].

Перевагою Airflow є підхід «workflows as code» – визначення пайплайнів у вигляді коду, а не у GUI-інструментах або XML-конфігураціях. Це надає кілька практичних переваг: пайплайни можна зберігати у системі контролю версій, тестувати та динамічно генерувати в залежності від вхідних умов [17]. Вбудований веб-інтерфейс відображає стан кожної задачі в реальному часі, дозволяє переглядати журнали виконання і перезапускати окремі задачі після усунення помилок, не перезапускаючи весь пайплайн.

Архітектурно Airflow складається з кількох компонентів: планувальника, який відстежує розклад і запускає задачі; виконавця, який визначає спосіб запуску задач – послідовно, паралельно на локальному вузлі (LocalExecutor) або розподілено через черги повідомлень (CeleryExecutor); веб-сервера для інтерфейсу; і метабази даних, де зберігається стан усіх запусків [18]. Для цього дослідження Airflow розгорнуто у Docker-контейнері з LocalExecutor, що є стандартним підходом для розробки і тестування.

Серед альтернатив Airflow варто зазначити Prefect і Dagster, які пропонують більш сучасний API і спрощений досвід розробки, але мають значно менший обсяг спільноти та екосистеми. Вибір Airflow для цього дослідження обумовлений зрілістю платформи, широкою підтримкою хмарних сервісів через провайдери (зокрема google-cloud-bigquery) і фактичним статусом галузевого стандарту оркестрації ETL-пайплайнів [17].

Google BigQuery є цільовою платформою зберігання і обробки даних у цьому дослідженні, її архітектурні характеристики розглянуто у підрозділі 1.2. З точки зору ролі у пайплайні BigQuery виступає єдиним сховищем для всіх трьох шарів: Raw, Staging і Mart. Завантаження сирих даних з усіх чотирьох джерел здійснюється через бібліотеку pandas-gbq з Airflow; трансформації Staging і Mart виконуються засобами dbt безпосередньо у BigQuery; готові аналітичні таблиці підключаються до Power BI у режимі імпорту.

Організація даних у BigQuery реалізована через датасети – логічні контейнери для таблиць і представлень. У проєкті використовуються три датасети: raw (сирі дані з джерел), analytics_staging (SQL-представлення шару Staging) і analytics (фінальні таблиці шару Mart). Така організація відповідає трирівневій архітектурі і дозволяє розмежувати відповідальність між шарами. Доступ до кожного датасету здійснюється через Identity and Access Management (IAM) Google Cloud, що забезпечує контроль над тим, хто і який рівень даних може читати або модифікувати [19].

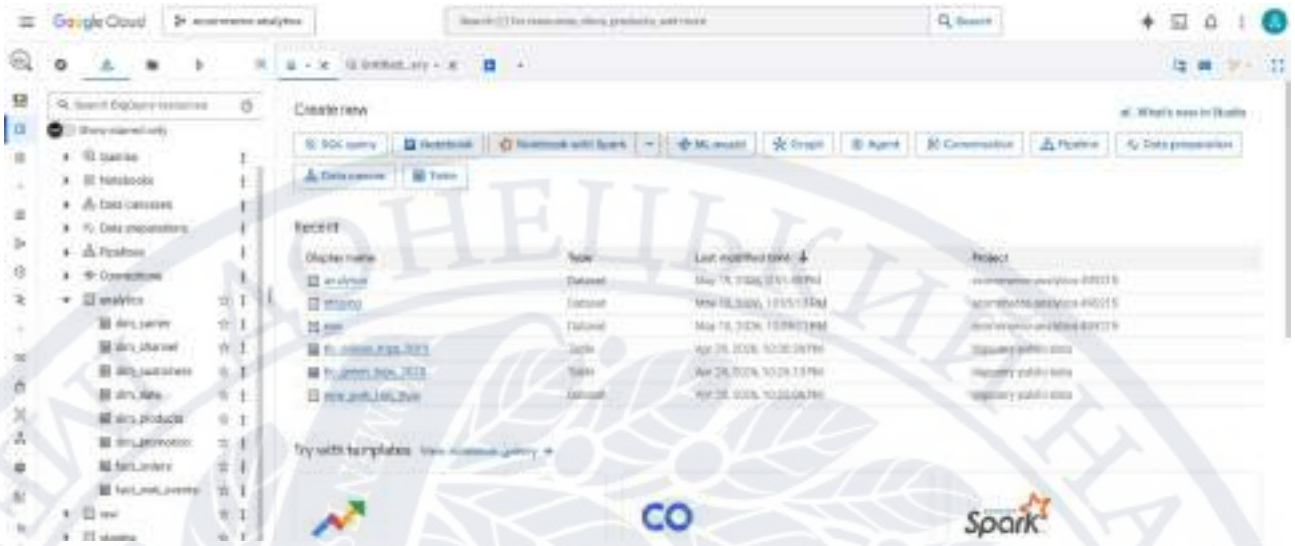

Рисунок 1.2 – Датасет analytics у Google BigQuery з вісьма аналітичними таблицями

dbt (data build tool) – інструмент трансформації даних, який дозволяє командам реалізовувати аналітичні перетворення засобами SQL з дотриманням практик програмної інженерії: модульністю, контролем версій, автоматизованим тестуванням і документацією [19]. Ключова ідея dbt полягає у тому, щоб розглядати дані так само, як розробники розглядають програмний код: кожна трансформація – окремий SQL-файл (модель), кожна модель може посилатися на інші через функцію {{ ref('model_name') }}, а dbt автоматично визначає порядок виконання на основі графу залежностей [19].

Проєкт dbt складається з кількох типів об'єктів:

1. Моделі – SQL SELECT-запити, що визначають трансформовану таблицю або представлення.
2. Джерела – посилання на сирі таблиці у сховищі, описані у YAML-файлах.
3. Макроси – повторно використовувані блоки Jinja-SQL-коду, аналоги функцій у програмуванні.
4. Тести – декларативні твердження про дані: унікальність ключів, відсутність пропусків, дотримання посилальної цілісності між таблицями [34]. Запуск команди dbt run виконує всі моделі у

правильному порядку; dbt test перевіряє якість даних після трансформації.

Для цього дослідження розроблено 38 dbt-моделей: 30 у шарі Staging і 8 у шарі Mart. Кожна staging-модель відповідає одній таблиці або джерелу і виконує уніфікацію типів та іменувань. Mart-шар формує схему «зірка» з двох таблиць фактів і шести вимірів. Макрос generate_schema_name дозволяє призначати власні імена датасетам у BigQuery замість стандартних.

Серед альтернатив dbt виділяють Apache Spark з SQL-трансформаціями, AWS Glue і власні збережені процедури у СУБД. Проте жоден з цих підходів не забезпечує такого рівня модульності, вбудованого тестування і генерації документації, як dbt. Саме поєднання орієнтованості на мову SQL і практик програмної інженерії робить dbt де-факто стандартом трансформацій у хмарних сховищах даних [19].

Microsoft Power BI – платформа бізнес-аналітики, яка входить до складу Microsoft Fabric і надає засоби для підключення до джерел даних, моделювання, обчислення показників і побудови інтерактивних звітів [20]. Вона складається з кількох компонентів:

1. Power BI Desktop – настільний застосунок для розробки моделей і звітів;
2. Power BI Service – хмарний сервіс для публікації і спільного використання;
3. Power BI Mobile – мобільний додаток для перегляду на пристроях.

Центральним поняттям моделі Power BI є семантична модель – набір таблиць, зв'язків між ними і DAX-мір. DAX (Data Analysis Expressions) – мова формул, яка дозволяє визначати обчислювані показники з урахуванням контексту фільтрів, застосованих до звіту [20]. Це принципово відрізняє DAX від звичайного SQL: міра автоматично адаптується до того, які елементи зрізу чи фільтра активовані у поточний момент. Наприклад, міра «Середній LTV» без додаткового коду повертає різне значення залежно від того, чи обраний сегмент «VIP», чи «Regular».

Для підключення до джерел даних Power BI підтримує три режими. Режим імпорту завантажує копію даних у внутрішній рушій VertiPaq і стискає їх у колонковому форматі – це забезпечує високу швидкість виконання запитів, оскільки всі обчислення відбуваються локально. Режим DirectQuery відправляє запит безпосередньо до джерела при кожній взаємодії з дашбордом – дані завжди актуальні, але швидкість залежить від продуктивності джерела. Режим Live Connection підключається до вже побудованої семантичної моделі, що зручно у великих організаціях з централізованою моделлю [20]. Для цього дослідження обрано режим імпорту, оскільки обсяг даних є невеликим, а швидкодія дашборду є пріоритетом.

Важливою особливістю Power BI як інструменту аналітики є крос-фільтрація: вибір елемента на будь-якій візуалізації автоматично фільтрує всі інші на тій самій сторінці звіту. Це дозволяє аналітику динамічно досліджувати дані – наприклад, клік на категорію «Електроніка» у гістограмі оновлює KPI-картки, кругову діаграму каналів і динаміку по місяцях. Функціональність деталізації дозволяє переходити з однієї сторінки на іншу з передачею контексту фільтра, що реалізовано у цьому проєкті через службову сторінку-підказку з деталізацією клієнтів за регіоном.

Вибір інструменту візуалізації для аналітичного пайплайну потребує порівняння наявних альтернатив. На ринку BI-платформ виділяють кілька інструментів, які претендують на схожі сценарії використання: Microsoft Power BI, Tableau (Salesforce), Looker Studio (Google) та Metabase. Tableau є визнаним лідером за гнучкістю візуалізації та глибиною аналітичних можливостей, але його ліцензія Creator коштує близько 75 дол. США на користувача на місяць, що робить його малодоступним для навчальних і малих проєктів. Looker Studio – безоплатний браузерний інструмент Google, нативно інтегрований з BigQuery та Google Analytics, однак його можливості моделювання даних і обчислення складних показників обмежені порівняно з Power BI. Metabase – відкрита платформа з безоплатним варіантом розгортання, орієнтована на прямий доступ

до бази даних, але без повноцінного шару семантичного моделювання і без підтримки складних формульних обчислень типу DAX [21].

Power BI поєднує широку підтримку джерел даних, потужну мову обчислень DAX, нативний конектор до Google BigQuery та безоплатну версію Desktop для розробки. Порівняно з Tableau він виграє за співвідношенням ціни і функціональності; порівняно з Looker Studio – за можливостями моделювання і обчислення бізнес-метрик; порівняно з Metabase – за наявністю семантичного шару і розвиненої системи ролей та рядкової безпеки [21]. Для цього проєкту визначальними чинниками стали: нативна інтеграція з BigQuery, можливість розрахунку метрик (маржинальність, LTV, конверсійна воронка) засобами DAX і безоплатна версія Desktop, що знімає будь-які ліцензійні обмеження на етапі розробки. Детальну порівняльну характеристику чотирьох BI-інструментів за основними критеріями наведено у додатку Л.

Детальну порівняльну характеристику інструментів пайплайну наведено у додатку М.

Кожен із чотирьох інструментів виконує визначену роль у пайплайні. Apache Airflow відповідає за оркестрацію і послідовне виконання задач вилучення та завантаження даних: він не зберігає і не трансформує дані, а лише координує запуск інших компонентів у правильному порядку з урахуванням залежностей. Google BigQuery є єдиним сховищем для всіх трьох шарів – Raw, Staging і Mart – і виконує SQL-запити, що лежать в основі трансформацій. dbt реалізує декларативний трансформаційний шар: кожна модель є SQL-файлом, результат якого матеріалізується у BigQuery; dbt відповідає виключно за T у підході ELT. Power BI підключається до mart-шару BigQuery в режимі імпорту, формує семантичну модель з DAX-мірами і надає кінцевим користувачам інтерактивний дашборд [21].

Таким чином, у першому розділі розглянуто теоретичні засади, на яких ґрунтується це дослідження: роль Business Intelligence і концепцію інтеграції різнорідних джерел, порівняльний аналіз архітектурних підходів до зберігання і обробки аналітичних даних, а також функціональні характеристики кожного з

обраних інструментів. Сформована теоретична база є підґрунтям для проєктування аналітичної архітектури і реалізації пайплайну, що описуються у наступних розділах.

Висновок до першого розділу

У першому розділі сформовано теоретичне підґрунтя для проєктування та реалізації аналітичного пайплайну на основі хмарного технологічного стеку.

Дослідження концепції Business Intelligence підтвердило, що BI є комплексною методологічною системою, яка охоплює не лише програмні засоби, а й процеси, архітектури та технології перетворення сирих даних на актуальну аналітичну інформацію. Ключовою функцією BI є забезпечення керівників єдиним поглядом на бізнес-показники через інтерактивні дашборди, що дозволяє замінити інтуїтивне управління доказовим підходом. Для підприємств електронної комерції BI є особливо актуальним, оскільки транзакційний характер бізнесу генерує значні обсяги розрізнених структурованих і напівструктурованих даних, ефективний аналіз яких потребує їх попередньої інтеграції.

Аналіз концепції інтеграції різнорідних джерел показав, що основними класами проблем при об'єднанні гетерогенних даних є структурна, семантична і технічна гетерогенність. Обґрунтовано доцільність підходу ELT (Extract, Load, Transform) порівняно з традиційним ETL: завантаження сирих даних у хмарне сховище до їх трансформації дозволяє зберегти початкові дані незмінними і повторити будь-який крок обробки без повторного звернення до джерел. Як концептуальна основа організації аналітичних таблиць обрано методологію Ральфа Кімбала – схему «зірка», яка забезпечує оптимальну продуктивність агрегацій і зрозумілість моделі для BI-інструментів.

Порівняльний аналіз класичних OLAP-систем і хмарних аналітичних сховищ підтвердив перевагу останніх для сценаріїв з динамічними вимогами до схеми і різнорідними джерелами. Колонково-орієнтоване зберігання,

розмежування обчислень і зберігання, serverless-модель та нативна підтримка SQL-трансформацій визначили Google BigQuery як цільову платформу. Серед розглянутих альтернатив – Snowflake, Amazon Redshift і Azure Synapse Analytics – BigQuery вирізняється безоплатним рівнем, що повністю покриває обчислювальні потреби проєкту, та нативною підтримкою завантаження структурованих, JSON і файлових джерел.

Огляд технологічного стеку підтвердив функціональну комплементарність чотирьох обраних інструментів. Apache Airflow забезпечує оркестрацію та послідовне виконання задач вилучення й завантаження даних. Google BigQuery слугує єдиним сховищем для всіх трьох шарів пайплайну – Raw, Staging і Mart. dbt реалізує декларативний трансформаційний шар із вбудованим тестуванням якості і автоматичним управлінням залежностями. Power BI надає кінцевим користувачам інтерактивний дашборд з потужним шаром DAX-обчислень. Такий поділ відповідальності відповідає принципам Modern Data Stack і формує надійну архітектурну основу для побудови повноцінного аналітичного пайплайну, що розглядається у наступних розділах.

РОЗДІЛ 2
ПРОЄКТУВАННЯ АНАЛІТИЧНОЇ АРХІТЕКТУРИ ТА БАГАТОВИМІРНИХ МОДЕЛЕЙ ДАНИХ

2.1 Аналіз джерел даних та проєктування операційної бази даних e-commerce платформи

Проєктування аналітичної системи починається з аналізу даних, які підлягають обробці. Для e-commerce платформи центральним джерелом є операційна база даних, що фіксує всі транзакції в режимі реального часу. Від якості її проєктування залежить як надійність самого застосунку, так і повнота аналітичних даних, що надходитимуть у сховище.

Операційні бази даних для систем електронної комерції будуються на принципах OLTP (Online Transaction Processing) – підходу, оптимізованого для виконання великої кількості коротких транзакцій з гарантіями цілісності даних [1]. На відміну від аналітичних систем, де пріоритет надається швидкості читання і агрегації, OLTP-системи оптимізовані для запису: кожне замовлення, платіж чи зміна статусу фіксуються швидко і без втрат. Для досягнення цього OLTP-схеми нормалізуються – надлишкові дані виносяться в окремі таблиці, між якими встановлюються зв’язки [22].

Нормалізація бази даних – це процес організації таблиць і зв'язків між ними з метою усунення надлишковості і запобігання аномаліям при вставці, оновленні та видаленні даних. Третя нормальна форма, запропонована Едгаром Коддом у 1971 році, є найпоширенішим стандартом для БД: вона вимагає, щоб кожен неключовий атрибут таблиці залежав виключно від первинного ключа і не залежав транзитивно від інших неключових атрибутів [23]. На практиці це означає, що назва міста не зберігається безпосередньо у таблиці замовлень, а виноситься в окрему таблицю міст, на яку замовлення посилається через зовнішній ключ.

Для e-commerce платформи цього дослідження спроєктовано базу даних PostgreSQL, що складається з 27 таблиць у третій нормальній формі. Таблиці згруповано за дев'ятьма функціональними доменами. Персональні дані суб'єктів винесено в спільну сутність-суперклас person, яка зберігає повне ПІБ (ім'я, прізвище, по батькові), дату народження і стать, а контактну інформацію (email, телефон) – у таблицю person_contacts; на person посилаються і клієнти, і працівники, що дозволяє ідентифікувати одну фізичну особу незалежно від її ролі та уникнути дублювання персональних даних між доменами. Домен клієнтів містить таблиці customers (роль клієнта з посиланням на person) і customer_addresses – адреси доставки з прив'язкою до географічного довідника. Таке розмежування дозволяє одному клієнту мати кілька адрес без дублювання основного запису, а одній особі – виступати одночасно і клієнтом, і працівником. Домен географії включає таблиці countries, regions і cities, що утворюють трирівневу ієрархію для адресації. Варто зазначити, що при генерації адрес тимчасово окуповані регіони виключено зі списку доступних для замовлень.

Домен каталогу товарів складається з п'яти таблиць. Ієрархія categories → subcategories охоплює 8 категорій і 24 підкатегорії. Таблиця products зберігає 154 товари з унікальними SKU у форматі «КАТЕГОРІЯ-БРЕНД-НОМЕР» (наприклад, ELEC-APL-001). Таблиця product_prices реалізує зберігання цінової історії: для кожного товару існує два записи – базова ціна 2023 року і скоригована ціна 2024 року. Це дозволяє при аналізі продажів використовувати ціну, що діяла на момент конкретного замовлення, а не поточну. Таблиця suppliers містить 18 постачальників.

Ядром схеми є домен замовлень, що складається з п'яти таблиць. Таблиця orders об'єднує замовлення з клієнтом, адресою, каналом продажу (Сайт, Мобільний застосунок, Маркетплейс), акцією та поточним статусом. Таблиця order_items зберігає позиції з фактичною ціною на момент продажу і маркером повернення. Окрема таблиця order_status_history веде повний журнал змін статусів з міткою часу – це забезпечує можливість аналізу часу переходу між етапами обробки замовлення. Життєвий цикл замовлення охоплює п'ять станів:

new, paid, shipped, delivered, cancelled. Таблиця order_managers фіксує прив'язку менеджера до замовлення.

Домени платежів і доставки організовані аналогічно: payments зберігає одну транзакцію на замовлення з прив'язкою до способу оплати (Банківська картка, Накладений платіж), shipments – відправлення з номером відстеження і часовими мітками відправлення та доставки, carriers – трьох перевізників. Домен акцій включає таблиці promotions (15 кампаній за 2023–2024 роки зі знижками від 10% до 35%) і promotion_products, що реалізує зв'язок «багато до багатьох» між акціями і товарами. Домен персоналу налічує три таблиці: roles, employees (25 співробітників з посиланням на person) і order_managers. Контактна інформація співробітників зберігається в тій самій таблиці person_contacts, що й контакти клієнтів, що забезпечує єдиний підхід до зберігання контактної інформації в усій схемі. Домен відгуків представлений однією таблицею reviews, що зберігає оцінку (1-5) і коментар клієнта до конкретної позиції доставленого замовлення.

Структуру зв'язків між таблицями ілюструє рисунок 2.1, що відображає повну ER-діаграму бази даних.

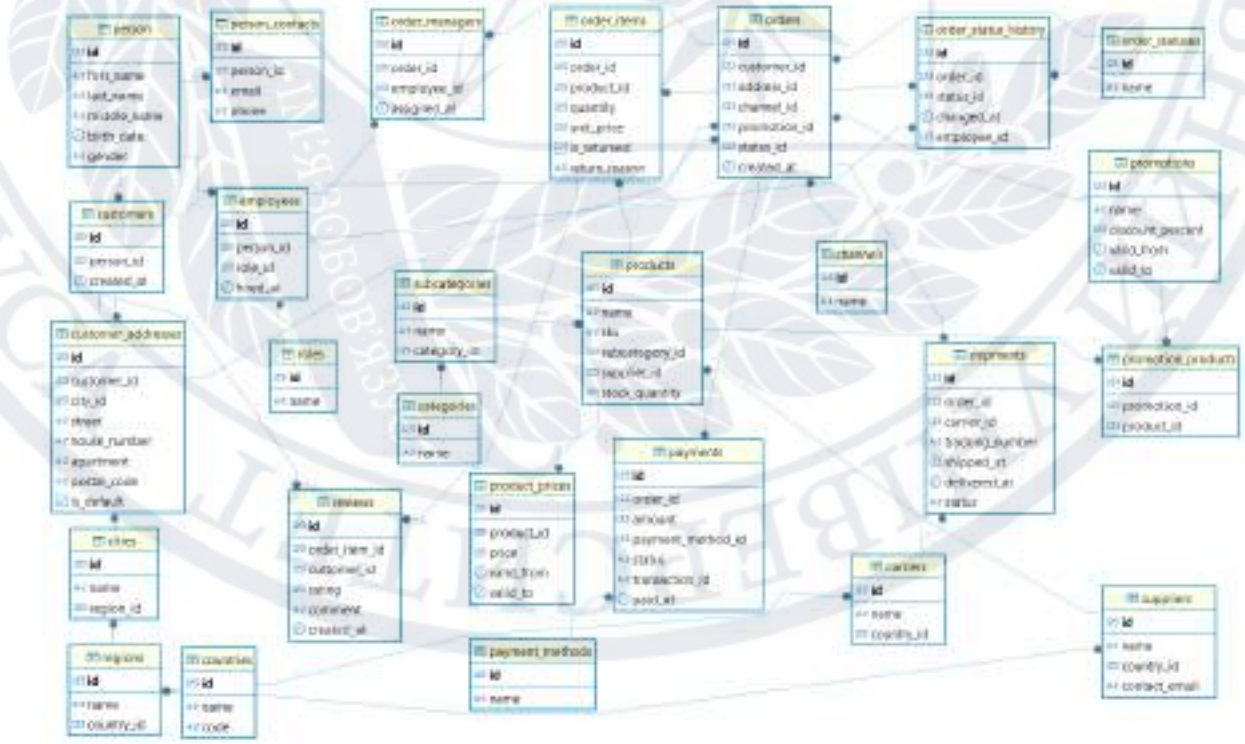

Рисунок 2.1 – ER-діаграма операційної бази даних (27 таблиць, 3НФ)

Для забезпечення цілісності даних усі зовнішні ключі визначено з явними стратегіями обробки при видаленні і оновленні батьківського запису. Для зв'язків, де дочірні записи не мають сенсу без батьківського, застосовано дію CASCADE – вона означає, що при видаленні батьківського запису всі пов'язані дочірні записи видаляються автоматично (наприклад, при видаленні замовлення автоматично видаляються всі його позиції). Для зв'язків, де видалення батьківського запису є небажаним, застосовано дію RESTRICT – вона блокує будь-яку спробу видалити батьківський запис, якщо на нього посилається хоча б один дочірній (наприклад, клієнта з наявними замовленнями видалити не вдасться, доки ці замовлення існують у базі) [24]. На всіх полях, що беруть участь у з'єднаннях і фільтрації, створено індекси – зокрема на зовнішніх ключах таблиць orders, order_items, payments, shipments і reviews. У PostgreSQL індекси на зовнішніх ключах не створюються автоматично, тому їх додано явно [26]. Офіційна документація PostgreSQL рекомендує явне визначення дій для підтримання посилальної цілісності (CASCADE, RESTRICT, SET NULL) для кожного зовнішнього ключа і додавання індексів на FK-полях у таблицях з великим обсягом операцій читання [25]. Практика явного індексування зовнішніх ключів є стандартною рекомендацією для OLTP-систем з активними операціями запису і JOIN-запитами [26].

Для автоматизації бізнес-логіки реалізовано два тригери. Тригер trg_decrease_stock спрацьовує після вставки запису в order_items і зменшує залишок товару на складі, попередньо перевіряючи його достатність. Тригер trg_log_status_change спрацьовує після оновлення статусу замовлення і автоматично додає запис до order_status_history. Додатково реалізовано збережену процедуру place_order, що інкапсулює повний цикл оформлення замовлення в одній транзакції: перевірку наявності товарів, створення замовлення, вставку позицій, розрахунок суми з урахуванням знижки і створення платіжного запису.

Окрім PostgreSQL, для аналітичного пайплайну підготовлено три зовнішніх джерела даних різних форматів, кожне з яких містить атрибути, відсутні в БД.

CSV-файл сегментації клієнтів (2 000 рядків) імітує вивантаження з CRM-системи. У реальних умовах саме CRM-система накопичує дані про взаємодію з клієнтами і автоматично формує подібні вивантаження; у цьому дослідженні цю роль імітує спеціально розроблений Python-скрипт. Він підключається до бази PostgreSQL, обчислює довічну цінність клієнта (сукупний дохід від усіх його завершених платежів) і на основі порогових значень присвоює сегмент: New (довічна цінність = 0), At-risk (до 10 000 грн), Regular (до 250 000 грн), VIP (250 000 грн і більше). Результуючий розподіл: 236 нових клієнтів, 254 під загрозою відтоку, 1 401 постійних і 109 VIP. Крім сегменту, файл містить канал залучення (Organic, Paid Search, Social Media, Referral, Email) і дату першої покупки – атрибути, яких немає в жодній таблиці PostgreSQL.

JSON-файл подій веб-аналітики (50 000 записів) відтворює потік поведінкових подій на сайті. Він згенерований окремим Python-скриптом, що формує події чотирьох типів із заданими ваговими коефіцієнтами: перегляд сторінки – 29 951 подія, додавання до кошика – 12 532, оформлення замовлення – 5 035, покупка – 2 482. Кожна подія містить унікальний ідентифікатор сесії, ідентифікатор клієнта (30% подій є анонімними), URL сторінки, тип пристрою і мітку часу в діапазоні 2023–2024 років. Ці дані є основою для розрахунку конверсійних метрик воронки продажів у mart-шарі.

XLSX-файл прайс-листа постачальника (154 рядки) згенеровано Python-скриптом, що вибирає всі товари з їх базовими цінами 2023 року і розраховує собівартість кожного як добуток ціни на випадковий коефіцієнт у діапазоні 0,55–0,70 – тобто постачальник отримує від 55% до 70% роздрібної ціни. Результат зберігається у форматі Excel із відформатованими заголовками стовпців, що імітує реальний прайс-лист. Поле собівартості є єдиним джерелом для розрахунку маржинальності товарів у аналітичній моделі – без нього обчислення показника маржинальності у вимірі товарів було б неможливим.

Таким чином, сформовано чотири різнорідних джерела даних, які разом охоплюють усі необхідні аналітичні виміри e-commerce платформи: транзакційну деталізацію, клієнтську сегментацію, поведінкові події та маржинальність товарів. Побудова інфраструктури для автоматизованого збору цих даних у єдине хмарне сховище розглядається у наступному підрозділі.

2.2 Побудова інфраструктури збору даних та ELT-пайплайну

Після проєктування бази даних і зовнішніх джерел постає завдання побудови інфраструктури, що автоматизує збір даних з усіх чотирьох джерел і завантажує їх у хмарне сховище. Для вирішення цього завдання застосовано Apache Airflow – платформу оркестрації, розгорнуту у Docker-середовищі, що забезпечує відтворюваність конфігурації і незалежність від локального оточення розробника [27].

Docker Compose дозволяє описати всі компоненти середовища в одному YAML-файлі і запустити їх однією командою. Для Airflow це означає, що веб-сервер, планувальник і база метаданих стартують узгоджено, а всі залежності – Python-бібліотеки, облікові дані Google Cloud, шляхи до файлів джерел – ізольовані всередині контейнерів і не конфліктують з іншим програмним забезпеченням на машині розробника [27]. Як виконавець задач обрано LocalExecutor, що дозволяє запускати задачі паралельно в межах одного вузла без розгортання розподіленої інфраструктури на основі черг повідомлень – таке рішення є стандартним для розробки і достатнім для обсягів даних цього проєкту [27].

Пайплайн складається з чотирьох DAG – по одному на кожне джерело даних. Такий підхід забезпечує незалежне виконання і моніторинг кожного пайплайну: відмова при завантаженні одного джерела не блокує завантаження решти.

Перший DAG відповідає за вивантаження всіх 27 таблиць операційної бази PostgreSQL у датасет raw Google BigQuery. Він складається з 27 задач, між якими

визначено явні залежності відповідно до структури зовнішніх ключів бази даних. Наприклад, задача завантаження міст виконується лише після завантаження регіонів, а задача завантаження замовлень – тільки після завантаження клієнтів, адрес, каналів і статусів. Кожна задача підключається до PostgreSQL через psycopg2, виконує SELECT усіх рядків таблиці, завантажує результат у DataFrame бібліотеки pandas і передає його до BigQuery, що замінює наявні дані у BigQuery при кожному запуску [28]. Такий режим повного перезавантаження обрано свідомо: обсяг таблиць є відносно невеликим і гарантує консистентність між BigQuery і PostgreSQL навіть у разі змін у джерелі. Підхід ELT набув широкого поширення з появою хмарних сховищ, що мають достатню обчислювальну потужність для SQL-трансформацій безпосередньо на сирих даних, усуваючи потребу у виділених серверах трансформації [29].

Другий DAG завантажує CSV-файл сегментації клієнтів у датасет raw. Задача зчитує файл через pandas і передає DataFrame до BigQuery. Третій DAG аналогічно обробляє JSON-файл веб-подій: він розбирає масив JSON-об'єктів у DataFrame, нормалізуючи вкладену структуру події у табличний формат, і завантажує результат у BigQuery. Четвертий DAG зчитує XLSX-файл прайс-листа через бібліотеку openpyxl і завантажує таблицю товарів з собівартостями у BigQuery [28].

Усі чотири DAG налаштовані без фіксованого розкладу і запускаються вручну. Таке рішення обумовлено статичним характером даних: вони не оновлюються в реальному часі, тому немає потреби в регулярному перезапуску. При необхідності перебудови сховища з нуля достатньо послідовно запустити всі чотири DAG в інтерфейсі Airflow. Варто зазначити, що у виробничому середовищі з динамічними даними ці DAG були б налаштовані на виконання за розкладом (наприклад, щоденно) або запускалися б за подією – при надходженні нового файлу або після завершення нічного вікна реплікації операційної бази. Однак для цілей цього дослідження ручний запуск є достатнім і дозволяє повністю контролювати момент оновлення даних у сховищі.

Рисунок 2.2 – Список пайплайнів в інтерфейсі Apache Airflow

На рисунку 2.3 зображено граф залежностей DAG для завантаження операційної бази даних. Граф відтворює ієрархію зовнішніх ключів операційної бази: таблиці-довідники, на які посилаються інші, не мають вхідних залежностей і виконуються першими. Таблиці, які містять зовнішні ключі на ці довідники, можуть стартувати лише після їх успішного завантаження – адже до появи запису у батьківській таблиці посилання на нього з дочірньої таблиці було б некоректним з точки зору цілісності даних. Таблиці транзакцій (orders, order_items, payments, shipments) розташовані на найнижчих рівнях графу, оскільки залежать від найбільшої кількості довідників.

Рисунок 2.3 – Граф залежностей DAG для завантаження операційної бази даних у BigQuery

Після виконання всіх чотирьох DAG датасет raw у BigQuery містить 30 таблиць: 27 таблиць з PostgreSQL і по одній таблиці для кожного з трьох зовнішніх файлових джерел. Цей датасет є «точкою відліку» для всіх наступних трансформацій – якщо на будь-якому наступному кроці виникне помилка, можна повторити трансформації з нуля без повторного звернення до джерел.

Таким чином, побудована інфраструктура пайплайну забезпечує автоматизоване і відтворюване завантаження даних з чотирьох різнорідних джерел у єдиний сирий шар хмарного сховища. Незалежність DAG, явні залежності між задачами і режим повного перезавантаження гарантують узгодженість даних у BigQuery з операційними джерелами. Наступний підрозділ присвячено проєктуванню багатовимірної моделі даних, яка будується на основі цього сирого шару.

2.3 Проєктування багатовимірної моделі даних та методологія підготовки даних

Після завантаження сирих даних у хмарне сховище постає завдання організації аналітичної моделі, яка буде безпосередньо використовуватися інструментом візуалізації. Для цього застосовано підхід Ральфа Кімбала – побудову схеми «зірка», де центральні таблиці фактів оточені таблицями вимірів [11]. Цей підхід є протилежністю нормалізації, характерної для OLTP-систем: виміри свідомо денормалізуються, тобто атрибути, розподілені по кількох таблицях операційної бази, об'єднуються в одну таблицю виміру. Така денормалізація знижує кількість з'єднань у аналітичних запитах і підвищує їх швидкість [30].

Схема «зірка» відповідає на питання «хто, що, коли, де і як» щодо вимірюваних бізнес-подій. Таблиці фактів зберігають числові показники і зовнішні ключі до вимірів; таблиці вимірів містять описові атрибути, за якими ці показники аналізуються [30]. Важливою властивістю добре спроєктованої схеми «зірка» є конформованість вимірів: якщо один вимір використовується кількома

таблицями фактів, він є спільним для всіх них. У цій моделі виміри dim_date і dim_products є конформованими – вони пов'язані як з fact_orders, так і з fact_web_events, що дозволяє будувати наскрізні звіти, які поєднують транзакційні показники і показники веб-поведінки в єдиному аналітичному контексті [31]. Концепцію конформованих вимірів детально описано у методології Кімбала: виміри, що використовуються у кількох таблицях фактів, мають мати ідентичну структуру і бізнес-визначення, інакше порівняння між фактами стає неможливим [32].

Модель даних побудована за принципом сузір'я фактів – різновиду схеми «зірка» з кількома таблицями фактів і спільними вимірами. Таблиця фактів fact_orders містить 12 519 записів і є центральним елементом моделі. Кожен запис відповідає одній позиції замовлення і включає кількість товару, ціну одиниці, загальну суму рядка, розмір знижки, чисту суму, кількість днів доставки, статус замовлення і статус платежу. Вона пов'язана з усіма шістьма вимірами: dim_date (через date_key), dim_customers (через customer_key), dim_products (через product_key), dim_channel (через channel_key), dim_promotion (через promotion_key) і dim_carrier (через carrier_key). Таблиця фактів fact_web_events містить 50 000 записів подій веб-аналітики і пов'язана з вимірами дати та товару.

Вимір dim_customers є найбільш збагаченим: він об'єднує атрибути з семи staging-моделей. З операційної бази беруться ім'я та прізвище клієнта, контактний email і адреса; з CSV-джерела сегментації – маркетинговий сегмент, довічна цінність клієнта і канал залучення. Таке об'єднання неможливо реалізувати в операційній базі без порушення принципів нормалізації, але в аналітичному вимірі воно є природним – аналітик хоче бачити повний профіль клієнта в одній таблиці. Вимір dim_products аналогічно збагачений: до стандартних атрибутів (назва, SKU, категорія, постачальник) додається поле margin_percent, розраховане як різниця між ціною продажу і собівартістю з XLSX-прайсу, поділена на ціну продажу. Вимір dim_date генерується засобами dbt через date-spine – послідовність дат від 2023-01-01 до 2024-12-31 з

атрибутами дня, тижня, місяця (включно з українськими назвами), кварталу, року і ознакою вихідного дня. Виміри dim_channel, dim_promotion і dim_carrier є компактними довідниками: канали продажу, акційні кампанії і перевізники відповідно.

Схему «зірка» у вигляді подання моделі Power BI зображено на рисунку 2.4.

Рисунок 2.4 – Схема «зірка» у вкладці «Подання моделі» Power BI

Між сирим шаром і mart-шаром розташований шар Staging, що виконує нормалізацію і стандартизацію даних. Staging-моделі реалізовано як SQL-представлення у BigQuery, а не як матеріалізовані таблиці – це означає, що вони не зберігають дані фізично, а лише визначають логіку трансформації, яка

виконується динамічно при зверненні [19]. Така конфігурація є оптимальною для проміжного шару: вона не збільшує обсяг сховища і гарантує, що mart-шар завжди відображає актуальні дані з raw при кожному запуску dbt.

Кожна staging-модель виконує кілька стандартних операцій підготовки даних. По-перше, приведення типів: усі числові ідентифікатори приводяться до типу INT64, числові показники – до NUMERIC або FLOAT64, дати – до DATE або TIMESTAMP залежно від контексту. По-друге, уніфікація іменувань: технічні назви полів операційної бази (наприклад, id) перейменовуються на аналітично значущі (наприклад, order_id, customer_id), що усуває неоднозначність при з'єднанні кількох staging-моделей. По-третє, усунення технічних артефактів: зокрема, у staging-моделі акцій виконується відновлення апострофа в назві «Чорна п'ятниця», який некоректно передається при вивантаженні через особливості кодування. По-четверте, для staging-моделі веб-подій додається локалізоване поле event_type_ua з українськими відповідниками типів подій для відображення у дашборді.

Загалом staging-шар містить 30 представлень: по одному на кожну з 27 таблиць PostgreSQL і по одному для CSV- і JSON-джерел. XLSX-прайс постачальника не має окремої staging-моделі – його дані безпосередньо використовуються у mart-моделі dim_products. Таке рішення є свідомим відхиленням від суворого принципу «кожне джерело проходить через staging», виправданим тим, що прайс містить лише одне аналітично значуще поле (собівартість), яке не потребує нормалізації. Таким чином, кожна одиниця сирих даних проходить через чітко визначений шар нормалізації, перш ніж потрапити до mart-шару.

Важливим аспектом підготовки є забезпечення цілісності ключів між шарами. У mart-шарі замість первинних ключів операційної бази використовуються сурогатні ключі, що генеруються dbt через функцію {{ dbt_utils.generate_surrogate_key([...]) }}. Сурогатний ключ обчислюється як хеш від набору бізнес-атрибутів, що однозначно ідентифікують запис. Це гарантує стабільність ключів навіть у разі зміни числових ідентифікаторів у джерелі.

Оригінальні ідентифікатори з PostgreSQL (наприклад, order_id, customer_id) зберігаються в таблицях mart-шару поряд із сурогатними ключами, що забезпечує можливість аудиту і відстеження походження кожного запису аж до операційної бази.

Таким чином, багатовимірна модель даних, побудована за принципом схеми «зірка» з двома таблицями фактів і шістьма вимірами, забезпечує оптимальну основу для побудови аналітичного дашборду. Конформовані виміри дозволяють поєднувати показники з різних таблиць фактів, а шар Staging гарантує якість і узгодженість даних перед їх потраплянням до mart-шару. Технічна реалізація цієї моделі засобами dbt розглядається у наступному розділі.

Висновок до другого розділу

У другому розділі спроєктовано всю аналітичну інфраструктуру дослідження – від операційної бази даних до багатовимірної моделі mart-шару.

Операційну базу даних PostgreSQL для e-commerce платформи спроєктовано у третій нормальній формі і реалізовано у вигляді 27 таблиць, об'єднаних у дев'ять функціональних доменів: клієнти, географія, каталог товарів, замовлення, платежі, доставка, акції, персонал і відгуки. Нормалізація до 3НФ забезпечила усунення надлишковості і запобігання аномаліям при вставці, оновленні та видаленні даних. Посилальна цілісність підтримується через явно визначені стратегії CASCADE і RESTRICT для всіх зовнішніх ключів, а продуктивність JOIN-запитів оптимізовано шляхом явного індексування FK-полів. База містить понад 65 000 записів синтетичних транзакційних даних за 2023–2024 роки, що охоплюють повний операційний цикл: від оформлення замовлення до доставки і відгуку.

Для забезпечення повноти аналітичної моделі підготовлено три зовнішніх джерела різних форматів. CSV-файл сегментації клієнтів надає маркетинговий сегмент, довічну цінність і канал залучення – атрибути, відсутні в реляційній базі. JSON-файл веб-подій розкриває поведінкову воронку від перегляду

сторінки до покупки. XLSX-файл прайс-листа постачальника забезпечує собівартість товарів – єдине джерело для розрахунку маржинальності. Разом чотири різнорідних джерела охоплюють усі необхідні аналітичні виміри e-commerce: транзакційну деталізацію, клієнтську сегментацію, поведінкові події та маржинальність товарів.

Інфраструктуру збору даних реалізовано на основі Apache Airflow у Docker-середовищі з LocalExecutor. Чотири незалежні DAG-пайплайни забезпечують автоматизоване завантаження кожного джерела у датасет raw Google BigQuery. DAG для PostgreSQL містить 27 задач з явними залежностями, що відтворюють ієрархію зовнішніх ключів і гарантують коректний порядок завантаження таблиць. Режим повного перезавантаження при кожному запуску DAG забезпечує консистентність між BigQuery і операційними джерелами.

Багатовимірну модель mart-шару спроєктовано за принципом схеми «зірка» у варіанті сузір'я фактів: дві таблиці фактів оточені шістьма спільними вимірами. Конформованість вимірів дозволяє поєднувати показники з обох таблиць фактів в єдиному аналітичному контексті. Виміри dim_customers і dim_products збагачено атрибутами з зовнішніх джерел: перший об'єднує дані PostgreSQL і CSV-файлу сегментації, другий – PostgreSQL і XLSX-прайсу, що уможливлює розрахунок маржинальності безпосередньо у вимірі товарів. Шар Staging виконує нормалізацію і стандартизацію кожного джерела, гарантуючи якість і узгодженість даних перед їх потраплянням до mart-шару. Таким чином, у розділі сформовано вичерпну проєктну документацію, що визначає структуру, джерела та архітектуру аналітичної системи і слугує основою для її технічної реалізації.

РОЗДІЛ 3
РЕАЛІЗАЦІЯ АНАЛІТИЧНИХ МОДЕЛЕЙ ТА ВІЗУАЛІЗАЦІЯ БІЗНЕС-ПОКАЗНИКІВ

3.1 Технічне забезпечення консолідації даних у хмарному середовищі та розробка dbt-моделей

Після завантаження сирих даних у BigQuery і проєктування аналітичної моделі настає етап технічної реалізації трансформаційного шару. Для цього застосовано dbt – інструмент, що дозволяє визначати трансформації у вигляді модульних SQL-файлів з підтримкою залежностей, автоматичної документації і контролю якості даних. Підхід «трансформації як код» означає, що вся логіка перетворення даних є відтворюваною, модульною і піддається перегляду так само, як і програмний код [33].

Проєкт dbt організовано у два шари моделей. Шар Staging містить 29 представлень – по одному на кожне джерело даних. Кожна staging-модель є тонкою оберткою навколо відповідної сирої таблиці і виконує мінімальний набір стандартних перетворень: приведення типів, перейменування полів на аналітично значущі назви і усунення технічних артефактів конкретного джерела [33]. Шар Mart містить 8 матеріалізованих таблиць – 2 таблиці фактів і 6 вимірів, – що формують схему «зірка» для підключення до Power BI. На рисунку 3.1 зображено інтерфейс dbt docs з переліком усіх моделей проєкту, організованих за шарами.

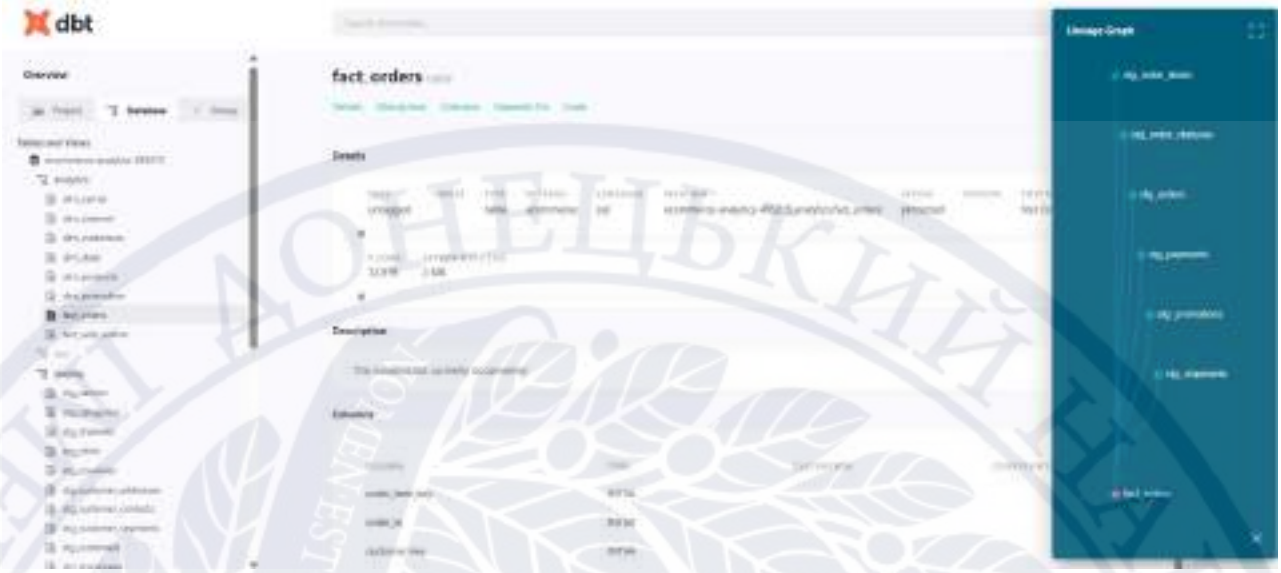

Рисунок 3.1 – Інтерфейс dbt docs з переліком моделей проєкту

Технічна реалізація staging-шару ґрунтується на кількох рішеннях. Усі staging-моделі матеріалізовано як представлення, що відповідає рекомендованій практиці: представлення не збільшують обсяг сховища і завжди відображають актуальні дані з raw-шару при виконанні mart-моделей [35]. Посилання між моделями реалізовано виключно через функцію {{ ref('model_name') }}, яка дозволяє dbt автоматично визначати порядок виконання моделей на основі графу залежностей і унеможливлює звернення до таблиць за жорстко заданими іменами. Посилання на сирі таблиці оформлено через {{ source('raw', 'table_name') }} з описом у sources.yml – це забезпечує єдине місце для управління всіма джерелами проєкту. dbt надає вбудовані тести якості даних: not_null перевіряє відсутність пропусків у ключових полях, unique – унікальність первинних ключів, accepted_values – допустимий набір значень для статусних полів [34].

Ряд staging-моделей потребував нестандартних рішень через особливості конкретних джерел. У моделі stg_promotions застосовано функцію REPLACE для відновлення апострофа в назві акції «Чорна п'ятниця», який некоректно передається при вивантаженні з PostgreSQL через особливості кодування символу. У моделі stg_web_events додано обчислюване поле event_type_ua з

українськими відповідниками типів подій через конструкцію CASE WHEN – це дозволяє відображати назви подій у дашборді українською без додаткових трансформацій у Power BI. Для dim_date реалізовано генерацію послідовності дат через макрос dbt date-spine з явним переліком українських назв місяців у полі month_name.

Особливістю mart-шару є те, що кожна модель вимірів і фактів будується виключно на staging-моделях, а не на сирих таблицях напряму. Це забезпечує чіткий розподіл відповідальності: staging відповідає за технічну нормалізацію, mart – за бізнес-логіку. Таблиця фактів fact_orders реалізована як JOIN шести staging-моделей з обчисленням чотирьох розрахункових полів: line_total (кількість × ціна одиниці), discount_amount (сума знижки від акції), net_amount (чиста сума після знижки) і days_to_deliver (різниця між датою доставки і датою замовлення у днях). Граф залежностей для fact_orders зображено на рисунку 3.2.

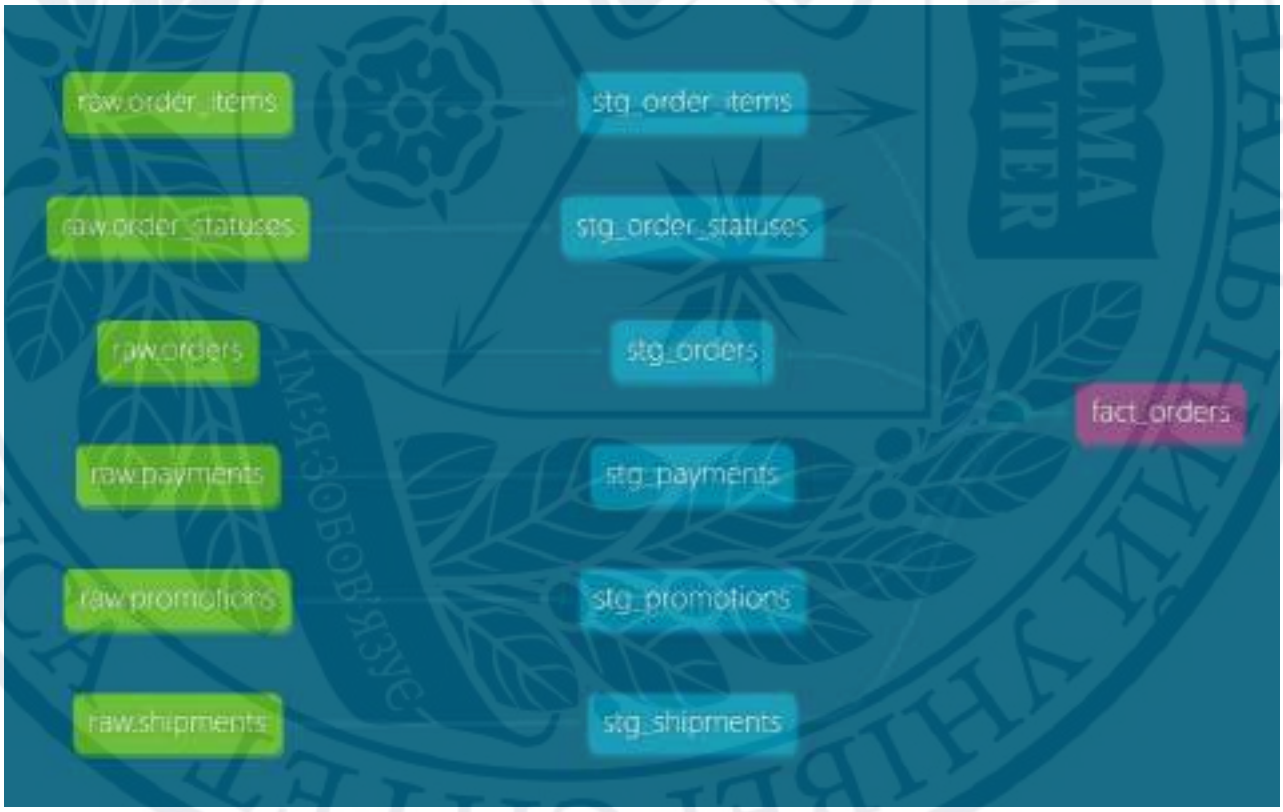

Рисунок 3.2 – Граф залежностей моделі fact_orders у dbt

На рисунку 3.2 показано трирівневу архітектуру: зелений шар raw (джерела), блакитний шар staging (нормалізовані представлення) і рожевий шар

mart (таблиця фактів). Кожна raw-таблиця проходить через відповідну staging-модель перед тим як потрапити до fact_orders – жодна mart-модель не посилається на raw напряму. Такий підхід гарантує, що зміна схеми у джерелі впливає лише на одну staging-модель, а не на всі mart-моделі, що від неї залежать.

Для підключення BigQuery до dbt використовується сервісний обліковий запис Google Cloud з ключем у форматі JSON. Параметри підключення описані у profiles.yml, який виключено з системи контролю версій через .gitignore. Ідентифікатор проєкту BigQuery і назви датасетів визначаються через змінні профілю і можуть бути змінені без правок у коді моделей. Кастомний макрос generate_schema_name перевизначає стандартну логіку dbt для призначення імен датасетів: замість дефолтного формату «profile_schema» моделі шару staging потрапляють у датасет analytics_staging, а моделі шару mart – у датасет analytics.

Запуск dbt run виконує всі 38 моделей у правильному порядку, визначеному графом залежностей. Час виконання повного циклу трансформацій складає близько 2–3 хвилин – це пов'язано з тим, що staging-моделі є представленнями і не потребують фізичного запису, тоді як mart-моделі матеріалізуються як таблиці і виконують повне перезавантаження при кожному запуску. Такий підхід відповідає принципам DataOps – набору практик, що поширюють принципи Agile і DevOps на аналітичну інженерію, зокрема автоматизацію тестів, відтворюваність пайплайнів і безперервний контроль якості даних [36]. Водночас це є рекомендованою практикою для невеликих наборів даних, де швидкість виконання є прийнятною, а пріоритетом залишається гарантія повної узгодженості між джерелом і mart-шаром [35].

Таким чином, технічна реалізація консолідації даних засобами dbt забезпечує модульну, відтворювану і контрольовану систему трансформацій. Перевірка якості даних на рівні пайплайну є критично важливою: помилки, не виявлені на ранніх етапах, поширюються через усі шари і призводять до некоректних аналітичних результатів [37]. Чіткий розподіл на staging і mart шари, використання ref() і source() для управління залежностями, а також кастомні рішення для локалізації і корекції артефактів джерел формують надійну основу

для побудови аналітичного дашборду. Реалізація бізнес-метрик і побудова інтерактивної звітності розглядаються у наступному підрозділі.

3.2 Реалізація бізнес-метрик та побудова інтерактивного дашборду в Power BI

Після завершення dbt-трансформацій аналітичні таблиці датасету analytics у BigQuery готові до підключення до Power BI. Підключення реалізовано через конектор Google BigQuery у режимі імпорту: Power BI завантажує копію даних у внутрішній рушій VertiPaq і стискає їх у власному колонковому форматі [20]. Microsoft рекомендує режим імпорту для наборів даних обсягом до 1 ГБ, де пріоритетом є швидкість виконання запитів, і DirectQuery для великих або часто оновлюваних наборів, де актуальність даних важливіша за швидкодію [39]. Для датасету цього проєкту режим імпорту є оптимальним: всі обчислення виконуються локально без мережевих затримок, що забезпечує миттєву реакцію на взаємодію з дашбордом. Аутентифікація виконується через сервісний обліковий запис Google Cloud з тими самими правами, що і dbt-профіль.

У Power BI Desktop імпортовано всі 8 таблиць датасету analytics: fact_orders, fact_web_events, dim_customers, dim_products, dim_date, dim_channel, dim_promotion і dim_carrier. Зв'язки між таблицями визначено відповідно до схеми «зірка»: кожна таблиця фактів пов'язана з вимірами через відповідні ключі типом «один до багатьох» (один запис у вимірі – багато у факті). Напрямок фільтрації встановлено від вимірів до фактів, що відповідає стандартній практиці для схеми «зірка» і запобігає неоднозначним результатам при крос-фільтрації.

Важливим елементом семантичної моделі є DAX-міри – обчислювані показники, що враховують контекст фільтрів, застосованих до звіту. На відміну від обчислюваних стовпців, які розраховуються один раз при завантаженні даних, міри обчислюються динамічно при кожній взаємодії з дашбордом. Це дозволяє одній мірі повертати різні значення залежно від того, який рік, категорія або сегмент клієнтів обраний у фільтрі. Для аналітичного дашборду розроблено

21 DAX-міру, організовану у сім тематичних папок. Нижче описано кожну з папок і логіку обчислення відповідних мір.

Папка Sales містить основні показники виручки і продажів. Центральною мірою є GMV (Gross Merchandise Value) – загальний обсяг продажів, що розраховується як сума добутків кількості та ціни одиниці для всіх позицій замовлень зі статусом, відмінним від cancelled. Функція SUMX виконує ітерацію по рядках таблиці фактів у поточному контексті фільтра, що дозволяє автоматично враховувати будь-які обрані фільтри: рік, категорію, канал продажу тощо. AOV (Average Order Value, середній чек) є похідною мірою і розраховується як відношення GMV до кількості замовлень.

Папка Margin містить показники маржинальності, можливість розрахунку яких з'явилася завдяки збагаченню виміру dim_products собівартістю з XLSX-джерела. Валовий прибуток обчислюється як сума добутків чистої суми замовлення на коефіцієнт маржинальності кожного товару. Середня маржинальність у відсотках розраховується як зважене середнє по всіх позиціях у поточному контексті фільтра.

Папка Efficiency містить показники операційної ефективності. Відсоток повернень розраховується через функцію DIVIDE як відношення кількості позицій з прапором is_returned до загальної кількості доставлених позицій. Функція DIVIDE використовується замість простого ділення, оскільки вона коректно обробляє ситуацію ділення на нуль, повертаючи 0 замість помилки. Вчасна доставка визначається як частка замовлень з терміном доставки не більше семи днів серед усіх доставлених.

Папка Time Intelligence містить міри для часового аналізу з використанням функцій DAX, які модифікують контекст фільтра по даті [38]. За визначенням SQLBI, функція CALCULATE є фундаментальною для будь-яких нетривіальних DAX-обчислень, оскільки лише вона дозволяє змінювати контекст фільтра всередині міри [40]. GMV YTD обчислюється через TOTALYTD і повертає накопичений обсяг продажів від початку року до поточної дати у контексті фільтра. GMV попереднього місяця і GMV попереднього року реалізовані через

CALCULATE з функціями DATEADD і SAMEPERIODLASTYEAR відповідно – вони зміщують поточний часовий контекст на один місяць або рік назад, дозволяючи будувати порівняльні діаграми без ручного налаштування фільтрів.

Папки Customers і Web Analytics містять показники клієнтської бази і веб-аналітики. Середній LTV розраховується як середнє значення поля lifetime_value з виміру dim_customers у поточному контексті – наприклад, при виборі сегменту VIP міра автоматично повертає середній LTV лише для цього сегменту. Конверсія веб-подій обчислюється як відношення кількості подій типу purchase до кількості подій типу page_view, що відповідає визначенню конверсії як частки відвідувачів, які здійснили покупку.

Дашборд складається з п'яти аналітичних сторінок і однієї службової сторінки-підказки. Кожна сторінка охоплює окремий аналітичний домен і містить набір взаємопов'язаних візуалізацій з підтримкою крос-фільтрації: вибір елемента на будь-якій візуалізації автоматично фільтрує всі інші на сторінці.

Перша сторінка «Огляд продажів» є стартовою і надає керівнику загальний погляд на бізнес. У верхньому рядку розміщено шість KPI-карток: GMV (165 млн грн), кількість замовлень (5 тис.), AOV (33 тис. грн), кількість клієнтів (2 тис.), відсоток повернень (7,0%) і середній термін доставки (6,5 днів). Нижче розміщено чотири взаємодоповнюючі візуалізації: горизонтальна гістограма виручки за категоріями товарів у спадному порядку, кругова діаграма розподілу виручки між каналами продажу, стовпчаста діаграма динаміки виручки по місяцях і кільцева діаграма розподілу замовлень за статусами. Сторінку зображено на рисунку 3.3.

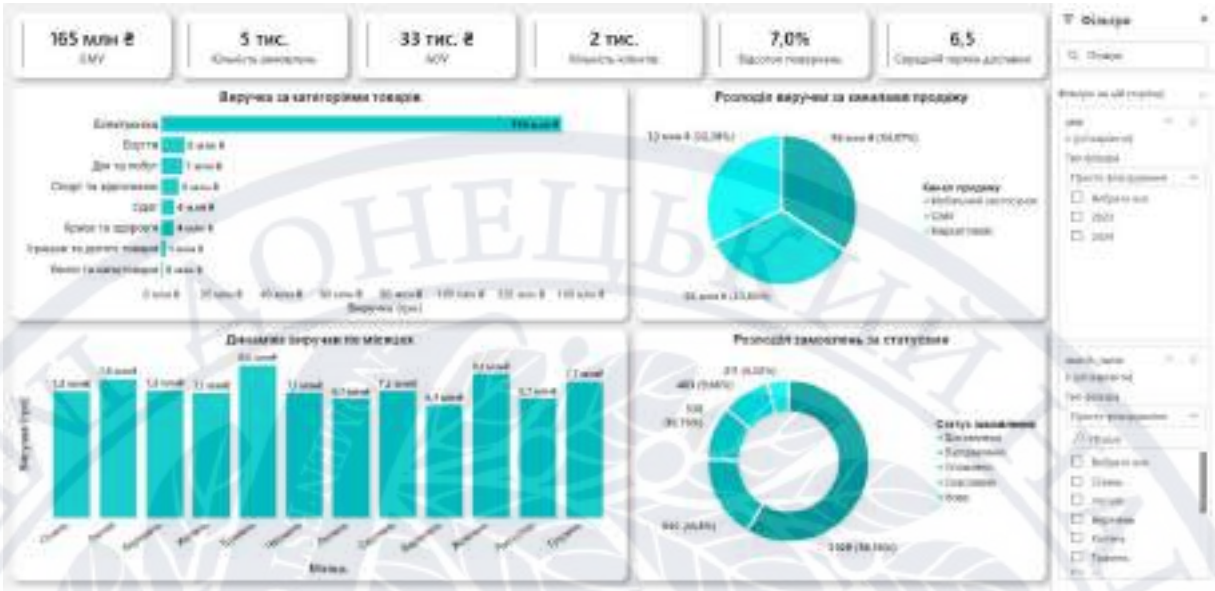

Рисунок 3.3 – Сторінка «Огляд продажів» аналітичного дашборду

Друга сторінка «Аналіз клієнтів» зосереджена на клієнтській базі і поведінці користувачів. У верхньому рядку чотири KPI-картки: кількість сесій (50 тис.), середній LTV (77,2 тис. грн), частка повторних покупок (76,4%) і конверсія (4,96%). Сторінка містить горизонтальну гістограму середнього LTV за сегментами клієнтів, гістограму кількості клієнтів за каналом залучення, кільцеву діаграму розподілу клієнтів за сегментами, воронку конверсії веб-подій і географічну карту Azure Maps з розподілом клієнтів по регіонах України. Сторінку зображено на рисунку 3.4.

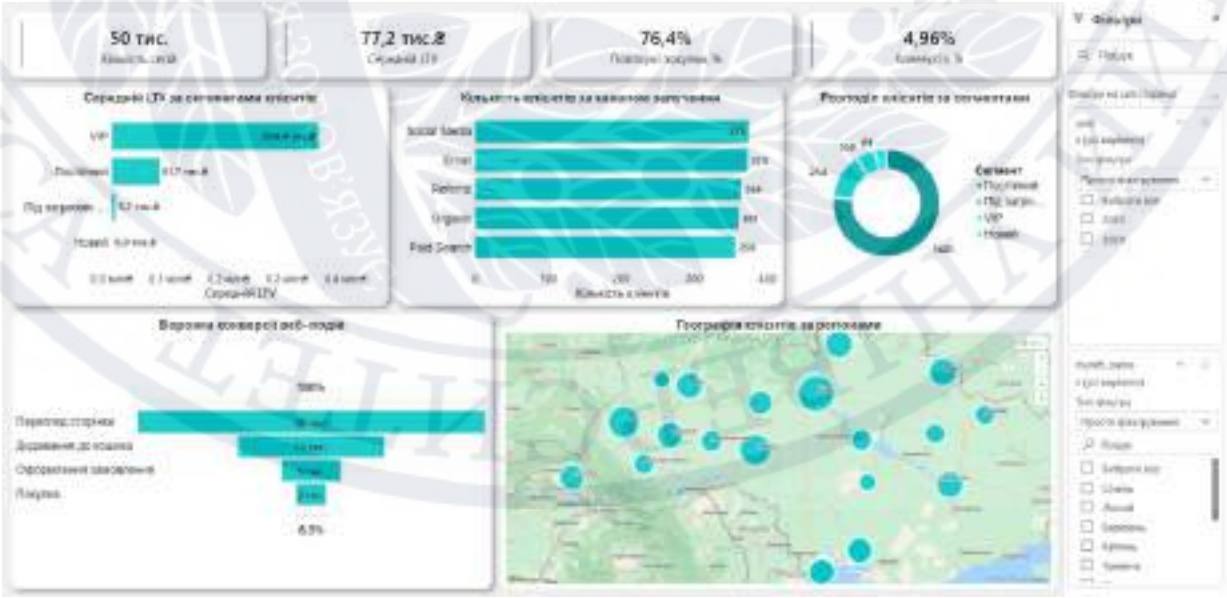

Рисунок 3.4 – Сторінка «Аналіз клієнтів» аналітичного дашборду

Механізм деталізації на карті дозволяє при наведенні на бульбашку регіону побачити підказку з деталізацією клієнтів по містах. На рисунку 3.5 показано приклад такої підказки для Київської області: 148 клієнтів розподілені по дев'яти містах, найбільше – у Фастові (26) та Борисполі (21).

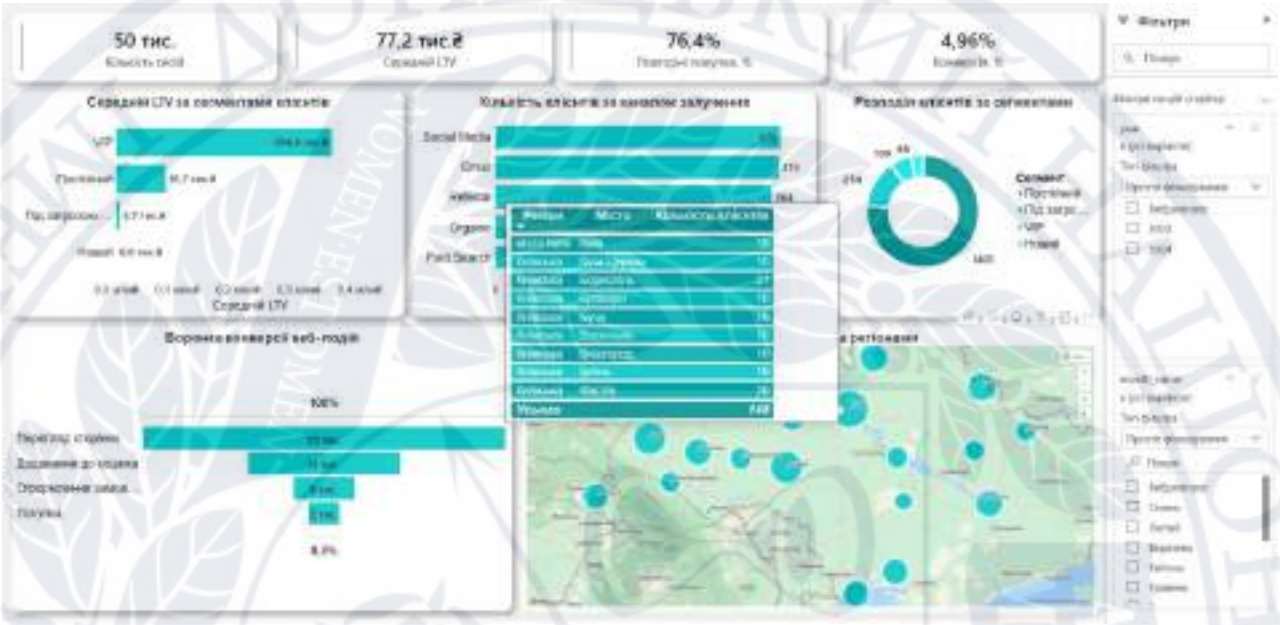

Рисунок 3.5 – Підказка з деталізацією клієнтів по містах Київської області

Третя сторінка «Аналіз товарів» надає детальний погляд на товарний портфель. KPI-картки відображають кількість проданих одиниць (23 тис.), середню маржинальність (44,55%), валовий прибуток (70 млн грн) і загальну суму знижок (3 млн грн). Горизонтальна гістограма топ-10 товарів за виручкою демонструє лідерство MacBook Pro 14 M3 (15,6 млн грн). Стовпчаста діаграма маржинальності за категоріями показує відносно рівномірний розподіл від 43,57% (Краса та здоров'я) до 45,45% (Одяг), що свідчить про збалансованість товарного портфелю з точки зору прибутковості. Діаграма відсотку повернень за категоріями і часова динаміка GMV доповнюють аналіз. Сторінку зображено на рисунку 3.6.

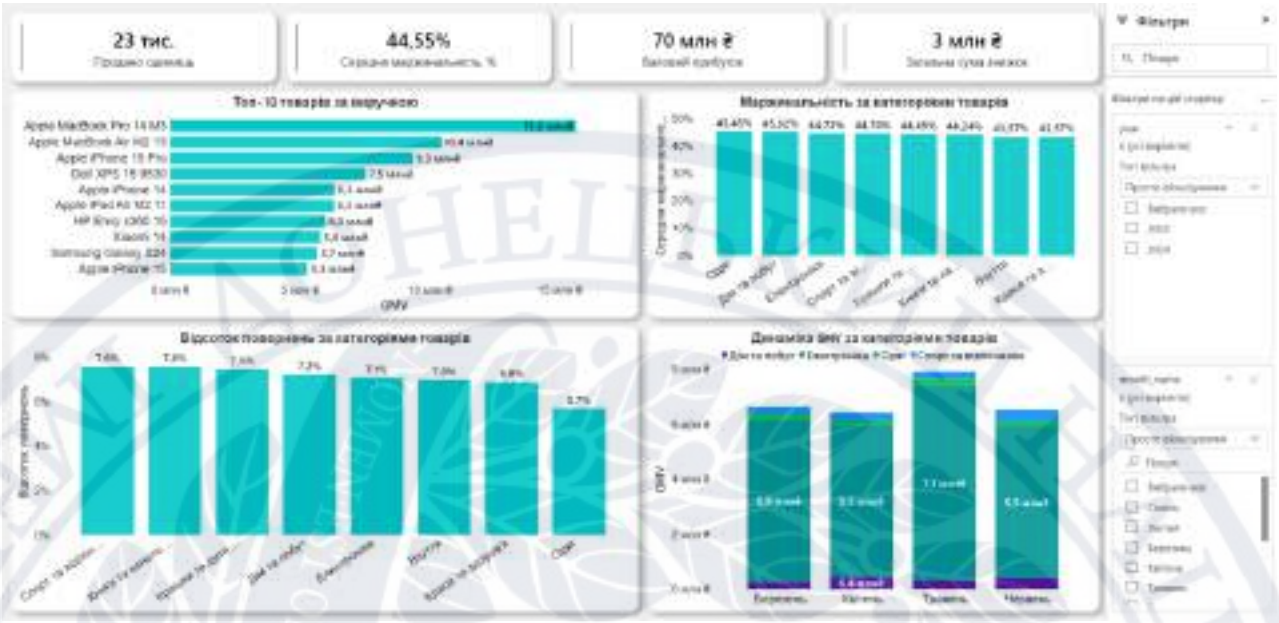

Рисунок 3.6 – Сторінка «Аналіз товарів» аналітичного дашборду

Четверта сторінка «Ефективність доставки» фокусується на операційних показниках. Три KPI-картки відображають вчасну доставку (93,8%), середній термін доставки (6,5 днів) і відсоток повернень (7,0%). Лінійний графік динаміки відсотку повернень по місяцях демонструє сезонні коливання в діапазоні від 5,6% до 8,4%. Лінійний графік динаміки середнього терміну доставки показує відносну стабільність показника протягом обох років. Горизонтальна гістограма відсотку повернень за категоріями дозволяє визначити проблемні товарні групи. Кільцева діаграма повернень за сегментами клієнтів розкриває цікаву закономірність: VIP-клієнти мають нижчий відсоток повернень (5,5%) порівняно з клієнтами під загрозою відтоку (7,6%). Сторінку зображено на рисунку 3.7.

Рисунок 3.7 – Сторінка «Ефективність доставки» аналітичного дашборду

П'ята сторінка «Часова аналітика» реалізує порівняльний аналіз між роками з використанням мір папки Time Intelligence. Лінійний графік GMV наростаючим підсумком з початку року порівнює траєкторії 2023 і 2024 років: у 2024 році GMV YTD стабільно перевищує показники 2023 року протягом усього року, досягаючи 88 млн грн до грудня проти 77 млн грн у 2023 році. Графік динаміки GMV по місяцях з двома рядами дозволяє виявити місяці з найбільшими відхиленнями між роками. Графік порівняння GMV поточного і попереднього місяця допомагає виявити короткострокові тенденції. Кільцева діаграма розподілу GMV по кварталах демонструє рівномірний розподіл виручки протягом року (Q1 – 24,32%, Q2 – 26,04%, Q3 – 24,3%, Q4 – 25,34%), що свідчить про відсутність яскраво вираженої сезонності у даних. Сторінку зображено на рисунку 3.8.

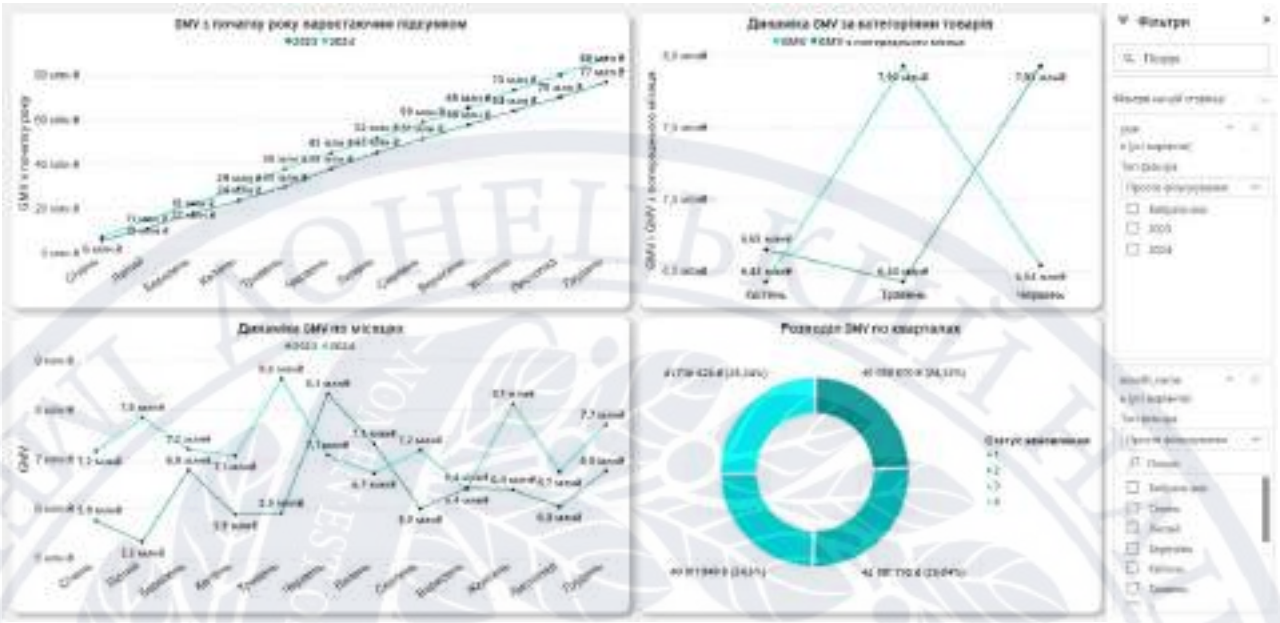

Рисунок 3.8 – Сторінка «Часова аналітика» аналітичного дашборду

Power BI дозволяє деталізувати будь-яку візуалізацію на окремому екрані. На рисунку 3.9 показано розгорнуту деталізацію графіку GMV наростаючим підсумком: при наведенні на точку липня відображається підказка з точними значеннями – 45 271 227 грн у 2023 році і 51 690 572 грн у 2024 році, що відповідає приросту 14,2% рік до року.

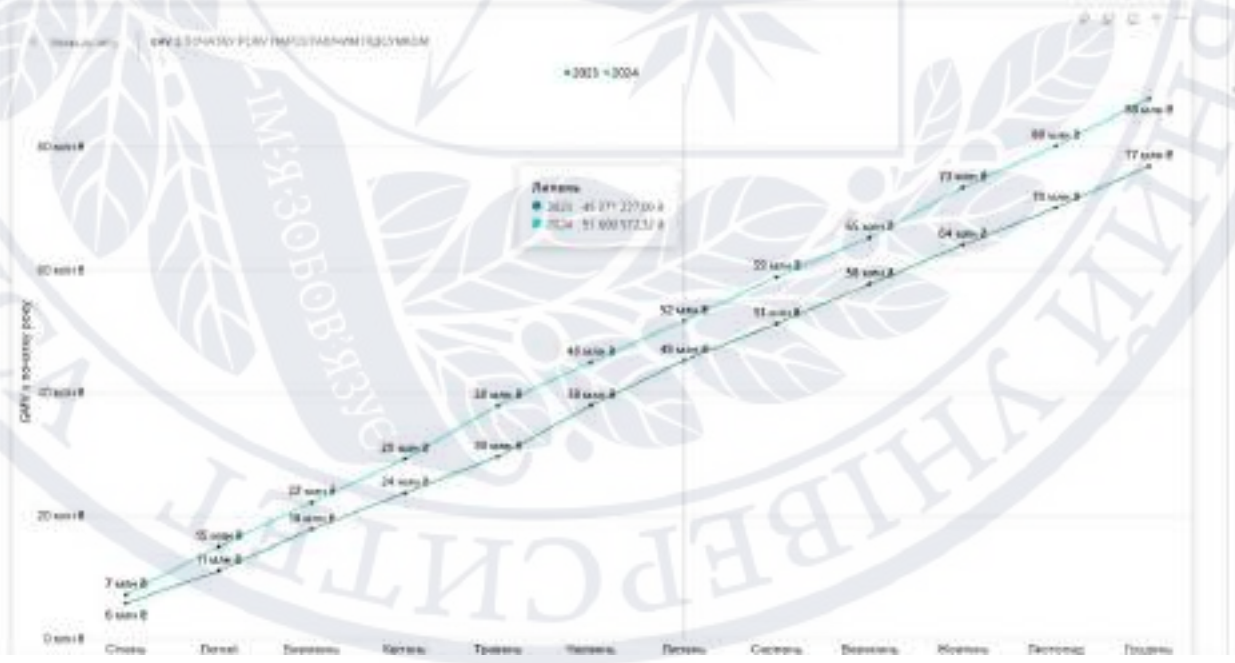

Рисунок 3.9 – Деталізація графіку GMV наростаючим підсумком з підказкою по липню

Аналогічний механізм деталізації реалізовано для географічної карти клієнтів. На рисунку 3.10 зображено повноекранну деталізацію карти з підказкою по Вінницькій області: 120 клієнтів розподілені по шести містах, найбільше – у Ладижині (25), Барі (21) і Вінниці (21). Кнопка «Назад до звіту» у верхньому лівому куті повертає користувача до основної сторінки «Аналіз клієнтів».

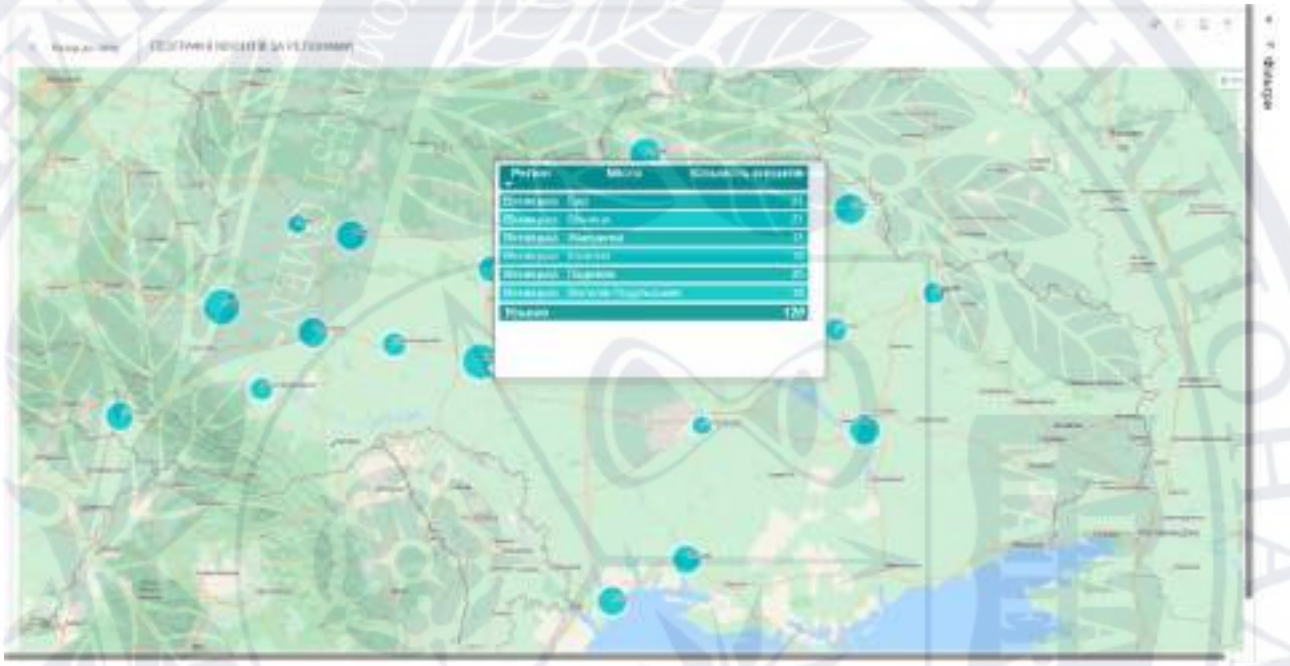

Рисунок 3.10 – Деталізація карти з підказкою по Вінницькій області

Окрім п'яти основних сторінок, реалізовано службову сторінку-підказку «Міста по регіону». Вона не відображається у навігаційному меню дашборду і викликається виключно через механізм деталізації при наведенні на регіон на географічній карті сторінки «Аналіз клієнтів». Підказка відображає таблицю з деталізацією кількості клієнтів по містах обраного регіону, що дозволяє аналітику переходити від регіонального рівня агрегації до рівня конкретних міст без переходу на іншу сторінку звіту.

Таким чином, реалізований аналітичний дашборд у Power BI забезпечує повний цикл аналізу e-commerce бізнесу: від загального огляду продажів до деталізації по сегментах клієнтів, товарному портфелю, операційній ефективності і часовій динаміці. Інтеграція даних з чотирьох різнорідних джерел

через dbt дозволила реалізувати показники, недосяжні при роботі лише з операційною базою даних. Інтерпретація отриманих аналітичних результатів наводиться у наступному підрозділі.

3.3 Оцінка адекватності моделі та інтерпретація результатів аналізу для прийняття рішень

Оцінка адекватності аналітичної моделі передбачає перевірку того, чи відповідають отримані показники закономірностям, закладеним при генерації синтетичних даних, і чи узгоджуються вони з галузевими орієнтирами для e-commerce платформ. Така перевірка є необхідною умовою для підтвердження коректності реалізованого пайплайну – від операційної бази до mart-шару і DAX-мір у Power BI.

Перевірка внутрішньої узгодженості даних здійснювалася на кількох рівнях. На рівні ETL перевірено, що кількість записів у raw-таблицях BigQuery відповідає кількості рядків у джерелах: 27 таблиць PostgreSQL завантажено повністю, CSV-файл містить рівно 2 000 записів, JSON-файл – 50 000 подій, XLSX-файл – 154 рядки. На рівні dbt-трансформацій перевірено, що fact_orders містить 12 519 записів, що відповідає кількості рядків у таблиці order_items операційної бази. На рівні Power BI перевірено, що сума GMV по всіх категоріях дорівнює загальному GMV на сторінці огляду, а сума замовлень за всіма статусами збігається із загальною кількістю замовлень. Ці перевірки підтверджують відсутність втрат і дублювань даних на кожному кроці пайплайну.

Додатковою перевіркою є порівняння розрахованих показників з параметрами генерації синтетичних даних. Відсоток повернень у дашборді становить 7,0%, що точно відповідає параметру is_returned=True для ~7% доставлених позицій, встановленому у скрипті генерації. Частка доставлених замовлень (58,56%) відповідає вазі статусу delivered (60% з невеликим відхиленням через випадковий розподіл). Розподіл між каналами продажу

близький до рівномірного (32–34% на канал), що відповідає рівновірогідному вибору каналу при генерації. Ці збіги підтверджують, що аналітична модель коректно відображає операційні дані без спотворень.

Наступним кроком є зіставлення отриманих показників з галузевими орієнтирами для e-commerce. За даними ECDB (2024), середній глобальний відсоток повернень у електронній комерції становить 10,1% [41]. Показник у 7,0%, отриманий у дашборді, є нижчим за середньогалузевий, що є позитивним сигналом і може свідчити про якісний клієнтський сервіс або точне відображення товарів. Водночас варто враховувати, що синтетичні дані генерувалися з фіксованим параметром 7%, тому це порівняння демонструє насамперед коректність моделі, а не реальну ефективність бізнесу.

Конверсія веб-подій становить 4,96%. За галузевими даними, середній глобальний показник конверсії e-commerce у 2024 році становить 2,7% [41]. Дослідження Triple Whale (2026) на основі 33 000 брендів підтверджує, що медіанна конверсія для більшості вертикалей становить 1,9–2,0%, а показник вище 3% свідчить про сайт з найвищою конверсією [43]. Отримане значення вище за середньогалузеве, що пояснюється особливостями синтетичної генерації: події purchase генерувалися з вагою 5%, а page_view – з вагою 60%, тоді як у реальних системах воронка конверсії значно крутіша через значну частку відмов на кожному кроці. Це є свідомим спрощенням для навчальних цілей і не впливає на коректність роботи пайплайну.

Частка повторних покупок становить 76,4%, що є високим показником. За даними Retalon (2026), середній рівень утримання клієнтів у роздрібній електронній комерції знаходиться в діапазоні 50–70% [42]. Підвищене значення у даному дослідженні пояснюється тим, що при генерації замовлень клієнти вибиралися рівноймовірно з пулу 2 000 осіб протягом двох років, що створює вищу частоту повторних покупок порівняно з реальною моделлю, де більшість клієнтів здійснюють лише одну покупку.

Середній чек (AOV) становить 33 000 грн (~800 USD за курсом на момент дослідження). За глобальними галузевими даними, AOV у e-commerce

коливається від 74 до 180 дол. США залежно від категорії [41]. Smart Insights вказує, що конверсійна воронка суттєво відрізняється залежно від того, як визначається знаменник – унікальні відвідувачі чи сесії, що є важливим при порівнянні даних між різними аналітичними системами [44]. Вищий показник у цьому дослідженні пояснюється домінуванням електроніки у товарному портфелі (134 млн грн з 165 млн – 81% виручки), яка традиційно має найвищий середній чек серед усіх категорій. Це підтверджується даними дашборду: гістограма топ-10 товарів показує, що дев'ять з десяти позицій є продуктами Apple, Dell або Xiaomi.

Результати аналізу дозволяють сформулювати ряд бізнес-інсайтів, які демонструють практичну цінність побудованого пайплайну.

Аналіз маржинальності за категоріями виявляє контрінтуїтивний результат: попри те, що електроніка домінує за виручкою (81%), її маржинальність (44,72%) не є найвищою серед категорій. Найвищу маржинальність демонструє одяг (45,45%) і дім та побут (45,32%). Це пояснюється меншою знижкою постачальника на електроніку (коефіцієнт собівартості ближчий до 0,70) порівняно з іншими категоріями. Для реального бізнесу такий аналіз міг би стати підставою для переоцінки асортиментної стратегії: диверсифікація продажів у бік категорій з вищою маржею при рівній виручці підвищила б загальний валовий прибуток.

Аналіз сегментів клієнтів розкриває значну диспропорцію у довічній цінності: VIP-клієнти (109 осіб, 5,45% бази) мають середній LTV 354,4 тис. грн проти 5,7 тис. грн у клієнтів під загрозою відтоку. Показово, що VIP-клієнти мають нижчий відсоток повернень (5,5%), ніж клієнти під загрозою відтоку (7,6%). Це підтверджує відому закономірність: лояльні клієнти з вищими витратами більш задоволені своїм досвідом. Для прийняття рішень такий результат вказує на пріоритетність програм утримання VIP-сегменту і заходів реактивації для сегменту At-risk.

Воронка конверсії веб-подій показує, що з 30 тис. переглядів сторінок до кошика доходить 13 тис. (43,3%), до оформлення – 5 тис. (38,5% від кошика), до

покупки – 2 тис. (40% від оформлення). Найбільший відсів відбувається на першому кроці (від перегляду до кошика), що відповідає типовій поведінці: більшість відвідувачів переглядають товари без наміру купити. Для реального бізнесу це вказувало б на необхідність покращення якості контенту на сторінках товарів.

Часова аналітика демонструє стабільне зростання GMV у 2024 році порівняно з 2023-м. У липні 2024 року GMV наростаючим підсумком становило 51,7 млн грн проти 45,3 млн грн у 2023 році – приріст 14,2%. Відсутність яскраво вираженої сезонності є характерною особливістю синтетичних даних, де замовлення генерувалися рівноймовірно протягом року. У реальних e-commerce системах спостерігається виражений сезонний пік у четвертому кварталі через Чорну п'ятницю і новорічні розпродажі.

При інтерпретації результатів необхідно враховувати обмеження, пов'язані із синтетичним характером даних. По-перше, рівномірний розподіл замовлень між каналами, категоріями і місяцями є наслідком детермінованої генерації, а не реальної ринкової динаміки. По-друге, маржинальність розрахована на основі фіксованих коефіцієнтів собівартості (0,55–0,70), тоді як у реальних системах вона залежить від обсягів закупівель, логістичних витрат і ринкової кон'юнктури. По-третє, відсутність реальних персональних даних виключає можливість ідентифікації конкретних клієнтських патернів. Ці обмеження не впливають на технічну коректність реалізованого пайплайну, але мають враховуватися при екстраполяції висновків на реальні бізнес-сценарії.

За класифікацією BigCommerce, показники дашборду охоплюють три з п'яти ключових KPI e-commerce: AOV, конверсію та LTV клієнтів, що є стандартним мінімальним набором для оцінки ефективності онлайн-торгівлі [45]. Таким чином, оцінка адекватності підтвердила коректність усіх компонентів аналітичного пайплайну: ETL-завантаження зберігає повноту і цілісність даних, dbt-трансформації правильно агрегують показники відповідно до бізнес-логіки, а DAX-міри у Power BI динамічно відповідають на зміну контексту фільтрів. Зіставлення з галузевими орієнтирами підтвердило, що

більшість показників знаходяться в очікуваних діапазонах з поясненими відхиленнями, зумовленими особливостями синтетичних даних. Побудований дашборд демонструє здатність інтегрованої аналітичної системи генерувати актуальні та обґрунтовані інсайти для підтримки прийняття бізнес-рішень.

Висновок до третього розділу

У третьому розділі реалізовано всі компоненти аналітичного пайплайну та підтверджено адекватність побудованої моделі.

Технічну реалізацію трансформаційного шару здійснено засобами dbt: розроблено 38 моделей, з яких 30 staging-представлень виконують нормалізацію кожного вхідного джерела, а 8 mart-таблиць формують схему «зірка». Staging-моделі матеріалізовано як представлення – це гарантує їх актуальність без дублювання даних у сховищі; mart-моделі матеріалізовано як таблиці для забезпечення максимальної продуктивності запитів з боку Power BI. Вбудовані dbt-тести (not_null, unique, accepted_values) автоматично перевіряють якість даних на кожному кроці трансформації. Нестандартні рішення – відновлення апострофа в назві акції через функцію REPLACE, генерація українських назв місяців через date-spine, додавання локалізованих типів веб-подій через CASE WHEN – забезпечили коректність і повноту аналітичних даних без перекладання трансформаційної логіки на Power BI.

Аналітичний дашборд у Power BI реалізовано з 21 DAX-мірою на п'яти тематичних сторінках: загальний огляд продажів, аналіз клієнтів, товарний аналіз, операційна аналітика і часова динаміка. Міри організовано у функціональні папки за типом показника – продажі і замовлення, повернення, маржинальність, клієнтська аналітика і веб-аналітика. Крос-фільтрація між усіма візуалізаціями забезпечує динамічний аналіз у довільному розрізі фільтрів, а функціональність деталізації реалізована через службову сторінку-підказку для аналізу клієнтів за регіоном.

Оцінку адекватності моделі проведено на трьох рівнях. На рівні внутрішньої узгодженості підтверджено відсутність втрат і дублювань: кількість записів у BigQuery відповідає джерелам, 12 519 записів fact_orders збігаються з таблицею order_items PostgreSQL, агрегати за всіма зрізами відповідають загальним підсумкам дашборду. На рівні відповідності параметрам генерації підтверджено, що показники дашборду точно відображають закладені значення: відсоток повернень – 7,0%, частка статусу delivered – ~58,56%, рівномірний розподіл між каналами продажу. На рівні зіставлення з галузевими орієнтирами встановлено, що відсоток повернень (7,0%) є нижчим за середньогалузевий рівень (10,1%), а відхилення AOV і конверсії від галузевих норм пояснюються особливостями синтетичних даних і не впливають на технічну коректність пайплайну.

Інтерпретація результатів виявила низку практично значущих бізнес-інсайтів. Попри домінування електроніки за виручкою (81% GMV), категорії одягу та товарів для дому демонструють вищу маржинальність, що відкриває простір для асортиментної оптимізації. Значна диспропорція в LTV між сегментами (VIP: 354 тис. грн проти At-risk: 5,7 тис. грн) обґрунтовує пріоритетність програм утримання лояльних клієнтів. Аналіз конверсійної воронки показав, що найбільший відсів відбувається на першому кроці – від перегляду до кошика, – що вказує на потенціал оптимізації контенту товарних сторінок. Часова аналітика зафіксувала зростання GMV у 2024 році порівняно з 2023-м на 14,2% наростаючим підсумком за сім місяців. Таким чином, реалізований пайплайн підтвердив здатність інтегрованої аналітичної системи генерувати достовірні і практично значущі бізнес-показники на основі різнорідних джерел даних.

ВИСНОВКИ

У кваліфікаційній роботі вирішено актуальне науково-практичне завдання – розроблено аналітичну платформу для e-commerce підприємства, яка реалізує повний цикл інтеграції різнорідних джерел даних і побудови інтерактивної бізнес-аналітики на основі сучасного технологічного стеку Apache Airflow – Google BigQuery – dbt – Power BI.

За результатами виконаної роботи зроблено такі висновки.

1. На підставі аналізу теоретичних засад бізнес-аналітики та концепції інтеграції різнорідних джерел даних обґрунтовано вибір трирівневої архітектури (Raw – Staging – Mart) і підходу ELT як методологічної основи побудови аналітичного пайплайну. Порівняльний аналіз класичних OLAP-систем і хмарних сховищ даних підтвердив перевагу колонково-орієнтованих serverless-платформ для сценаріїв з динамічними вимогами до схеми і різнорідними джерелами. Огляд і порівняльний аналіз сучасних BI-інструментів дозволив обґрунтувати вибір Power BI як платформи фінальної візуалізації завдяки нативній інтеграції з BigQuery, потужному мовному шару DAX і безкоштовній версії Desktop.

2. Спроєктовано операційну базу даних PostgreSQL e-commerce платформи у третій нормальній формі, що складається з 27 таблиць, об'єднаних у дев'ять функціональних доменів. База містить понад 65 000 записів синтетичних транзакційних даних за 2023–2024 роки і доповнена трьома зовнішніми джерелами: CSV-файлом сегментації клієнтів, JSON-файлом веб-подій і XLSX-файлом прайс-листа постачальника. Забезпечено посилальну цілісність через явно визначені стратегії CASCADE і RESTRICT для всіх зовнішніх ключів, а також індексування ключових полів для прискорення аналітичних запитів.

3. Реалізовано ETL-пайплайн на основі Apache Airflow, що складається з чотирьох незалежних DAG-пайплайнів для автоматизованого завантаження даних з усіх джерел у Google BigQuery. DAG для операційної бази містить 27 задач з явно визначеними залежностями, що відтворюють ієрархію зовнішніх

ключів і гарантують правильний порядок завантаження. Увесь пайплайн розгорнуто у Docker-середовищі з LocalExecutor, що забезпечує відтворюваність конфігурації і ізольованість від локального оточення.

4. Розроблено 38 dbt-моделей, що реалізують трирівневу аналітичну архітектуру: 30 staging-представлень для нормалізації та стандартизації даних кожного джерела і 8 mart-таблиць, що формують схему «зірка» за принципом сузір'я фактів. Схема містить дві таблиці фактів і шість вимірів. Виміри dim_customers і dim_products збагачено атрибутами з зовнішніх джерел: сегментом і довічною цінністю клієнта з CSV і маржинальністю товару з XLSX.

5. Побудовано інтерактивний аналітичний дашборд у Power BI з 21 DAX-мірою у семи тематичних папках і п'ятьма аналітичними сторінками: огляд продажів, аналіз клієнтів, аналіз товарів, ефективність доставки і часова аналітика. Реалізовано механізми деталізації і підказки для деталізації географічних і часових даних. Дашборд дозволяє аналізувати GMV, маржинальність, сегментацію клієнтів (VIP, Regular, At-risk, New) і конверсійну воронку веб-подій в єдиному аналітичному середовищі.

6. Проведено оцінку адекватності аналітичної моделі на трьох рівнях: внутрішня узгодженість даних між усіма шарами пайплайну, відповідність показників параметрам генерації синтетичних даних і зіставлення з галузевими орієнтирами e-commerce. Показник повернень (7,0%) є нижчим за глобальний середній (10,1%), частка вчасних доставок становить 93,8%, а середній LTV VIP-сегменту (354,4 тис. грн) у 4,3 раза перевищує показник постійних клієнтів. Оцінка підтвердила коректність реалізованого пайплайну і продемонструвала здатність системи генерувати обґрунтовані бізнес-інсайти для підтримки прийняття рішень.

Практична цінність роботи полягає у реалізації повного аналітичного пайплайну, що може бути відтворений і масштабований для реальних e-commerce підприємств. Запропонована архітектура забезпечує інтеграцію будь-якої кількості різнорідних джерел шляхом додавання нових DAG і

dbt-моделей без зміни існуючого коду. Результати роботи апробовано у двох наукових публікаціях.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Dimension Market Research. Business Intelligence Market Size to Reach USD 75.7 Billion by 2033 with 9.3% CAGR Growth. GlobeNewswire. 2024. URL: https://www.globenewswire.com/news-release/2024/11/25/2986708/0/en/Business-Intelligence-Market-Size-to-Reach-USD-75-7-Billion-by-2033-with-9-3-CAGR-Growth-Dimension-Market-Research.html.

2. Business intelligence. Wikipedia. URL: https://en.wikipedia.org/wiki/Business_intelligence.

3. What Is Business Intelligence (BI)? IBM. URL: https://www.ibm.com/think/topics/business-intelligence.

4. Business Intelligence Based on Big Data in Innovative E-commerce Data / DEAI 2024: Proceedings of the 2024 International Conference on Digital Economy and Artificial Intelligence. ACM, 2024. URL: https://doi.org/10.1145/3675417.3675463.

5. Zhao L. An In-Depth Analysis of Data Warehouse and Data Mining-Aligned Business Intelligence Tools and Methodologies in E-Commerce / MSIE 2024: Proceedings of the 6th International Conference on Management Science and Industrial Engineering. ACM, 2024. URL: https://doi.org/10.1145/3664968.3664991.

6. KPI Dashboards: Definition, Benefits, and Best Practices. Tableau. URL: https://www.tableau.com/kpi/what-is-kpi-dashboard.

7. Putrama I. M., Martinek P. Heterogeneous data integration: Challenges and opportunities. Data in Brief. 2024. Vol. 56. URL: https://doi.org/10.1016/j.dib.2024.110853.

8. 6 Data Integration Challenges & Solutions You Should Know. Airbyte. 2024. URL: https://airbyte.com/data-engineering-resources/data-integration-challenges.

9. Data Integration Challenges and Solutions. Dremio. URL: https://www.dremio.com/wiki/heterogeneous-data/.

10. What Is OLAP (Online Analytical Processing)? AWS. URL: https://aws.amazon.com/what-is/olap.

11. Star Schema in Data Warehouse Modeling. GeeksforGeeks. 2025. URL: https://www.geeksforgeeks.org/dbms/star-schema-in-data-warehouse-modeling/.

12. Kimball R. Dimensional Modeling Techniques. Kimball Group. URL: https://www.kimballgroup.com/data-warehouse-business-intelligence-resources/kimball-techniques/dimensional-modeling-techniques.

13. Snowflake vs. Amazon Redshift: Key Differences. Stitch. URL: https://www.stitchdata.com/resources/snowflake-vs-redshift/.

14. Azure Synapse Analytics documentation. Microsoft. URL: https://learn.microsoft.com/en-us/azure/synapse-analytics.

15. BigQuery: Cloud Data Warehouse. Google Cloud. URL: https://cloud.google.com/bigquery.

16. BigQuery pricing. Google Cloud. URL: https://cloud.google.com/bigquery/pricing.

17. Apache Airflow Documentation. Apache Software Foundation. URL: https://airflow.apache.org/docs.

18. Airflow Architecture Overview. Apache Software Foundation. URL: https://airflow.apache.org/docs/apache-airflow/stable/core-concepts/overview.html.

19. dbt Documentation. dbt Labs. URL: https://docs.getdbt.com.

20. Power BI documentation. Microsoft Learn. URL: https://learn.microsoft.com/en-us/power-bi.

21. Power BI vs Tableau vs Looker Studio vs Metabase. ikemo.io. URL: https://ikemo.io/blog/power-bi-vs-tableau-vs-looker-vs-metabase.

22. How to Design a Relational Database for E-commerce Website. GeeksforGeeks. 2025. URL: https://www.geeksforgeeks.org/dbms/how-to-design-a-relational-database-for-e-commerce-website/.

23. Database Normalization. GeeksforGeeks. URL: https://www.geeksforgeeks.org/normal-forms-in-dbms.

24. PostgreSQL 16 Documentation: Constraints. The PostgreSQL Global Development Group. URL: https://www.postgresql.org/docs/current/ddl-constraints.html.

25. PostgreSQL 16 Documentation: Referential Integrity. The PostgreSQL Global Development Group. URL: https://www.postgresql.org/docs/current/ddl-constraints.html#DDL-CONSTRAINTS-FK.

26. PostgreSQL Index Documentation. The PostgreSQL Global Development Group. URL: https://www.postgresql.org/docs/current/indexes.html.

27. Running Airflow in Docker. Apache Software Foundation. URL: https://airflow.apache.org/docs/apache-airflow/stable/howto/docker-compose/index.html.

28. pandas-gbq: Google BigQuery connector for pandas. Google. URL: https://pandas-gbq.readthedocs.io.

29. ELT vs ETL: What's the Difference? Airbyte. URL: https://airbyte.com/blog/elt-vs-etl.

30. Star Schema in Data Warehouse. Databricks. URL: https://www.databricks.com/glossary/star-schema.

31. Star schema and the importance of it. Microsoft Learn. URL: https://learn.microsoft.com/en-us/power-bi/guidance/star-schema.

32. Conformed Dimensions. Kimball Group. URL: https://www.kimballgroup.com/data-warehouse-business-intelligence-resources/kimball-techniques/dimensional-modeling-techniques/conformed-dimensions.

33. dbt Best Practices. dbt Labs. URL: https://docs.getdbt.com/best-practices.

34. dbt Testing. dbt Labs. URL: https://docs.getdbt.com/docs/build/data-tests.

35. dbt Materializations. dbt Labs. URL: https://docs.getdbt.com/docs/build/materializations.

36. DataOps: The Definitive Guide to Streamlining Data Pipelines. Airbyte. 2023. URL: https://airbyte.com/data-engineering-resources/dataops-the-definitive-guide.

37. Data Validation in ETL. Integrate.io. 2026. URL: https://www.integrate.io/blog/data-validation-etl.

38. DAX function reference. Microsoft Learn. URL: https://learn.microsoft.com/en-us/dax/dax-function-reference.

39. Power BI Desktop: DirectQuery and Import modes. Microsoft Learn. URL: https://learn.microsoft.com/en-us/power-bi/connect-data/desktop-directquery-about.

40. CALCULATE function (DAX). SQLBI. URL: https://dax.guide/calculate.

41. Global eCommerce Benchmarks: AOV, Conversion rate, Return rate. ECDB. 2024. URL: https://ecommercedb.com/benchmarks/ww/all.

42. 10 Most Important Ecommerce KPIs to Measure in 2026. Retalon. URL: https://retalon.com/blog/ecommerce-kpis.

43. Ecommerce Benchmarks 2025: Key Metrics & Industry Data. Triple Whale. 2026. URL: https://www.triplewhale.com/blog/ecommerce-benchmarks.

44. Ecommerce conversion rates. Smart Insights. 2024. URL: https://www.smartinsights.com/ecommerce/ecommerce-analytics/ecommerce-conversion-rates/.

45. Ecommerce Metrics in 2026 (KPIs to Measure Your Success). BigCommerce. URL: https://www.bigcommerce.com/articles/ecommerce/ecommerce-metrics/.

ДОДАТКИ

ДОДАТОК А

Зображення аналітичного дашборду Power BI в режимі інтерактивної фільтрації

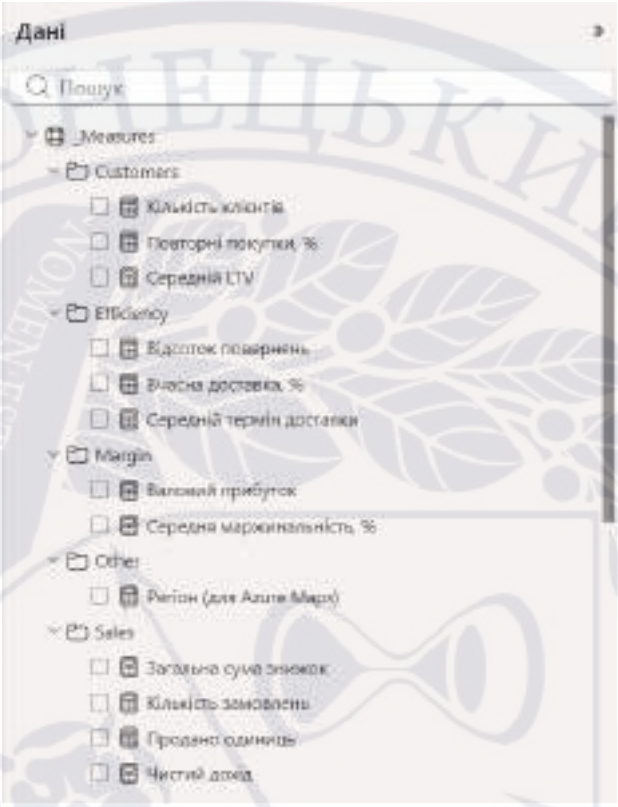

Рисунок А.1 – Фрагмент переліку DAX-мір аналітичного дашборду у панелі даних Power BI

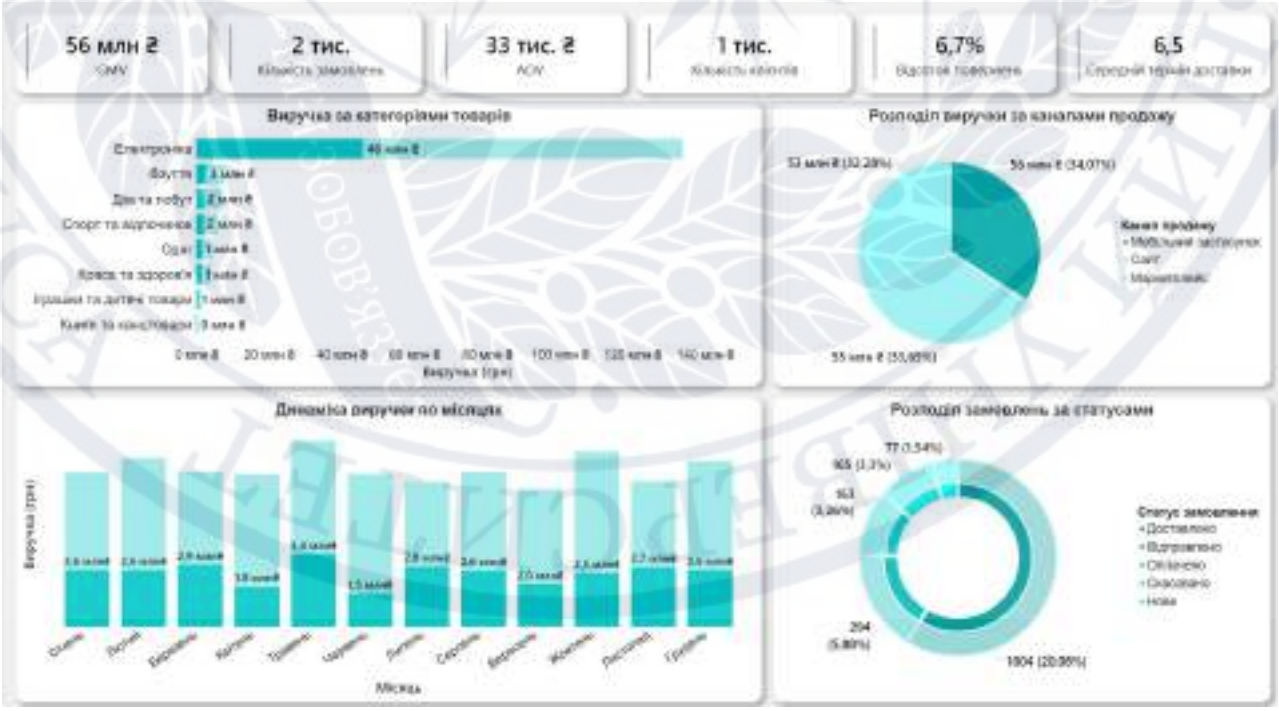

Рисунок А.2 – Крос-фільтрація за каналом продажу «Мобільний застосунок»

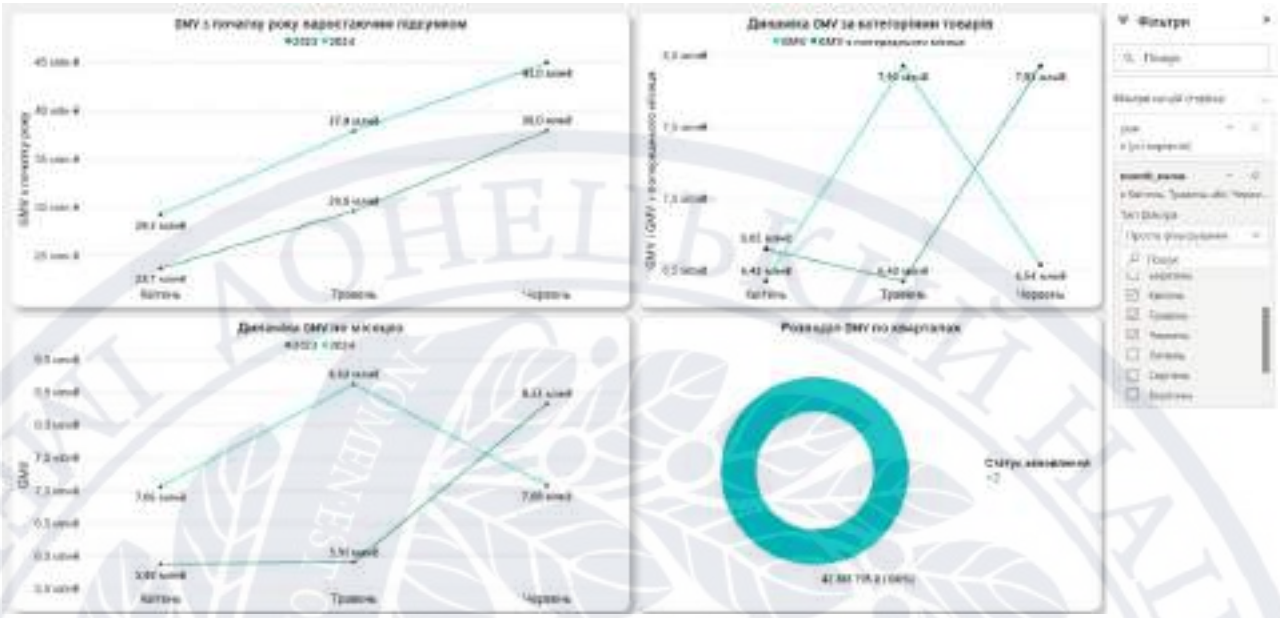

Рисунок А.3 – Фільтрація даних за квітень–червень у сторінці «Часова аналітика»

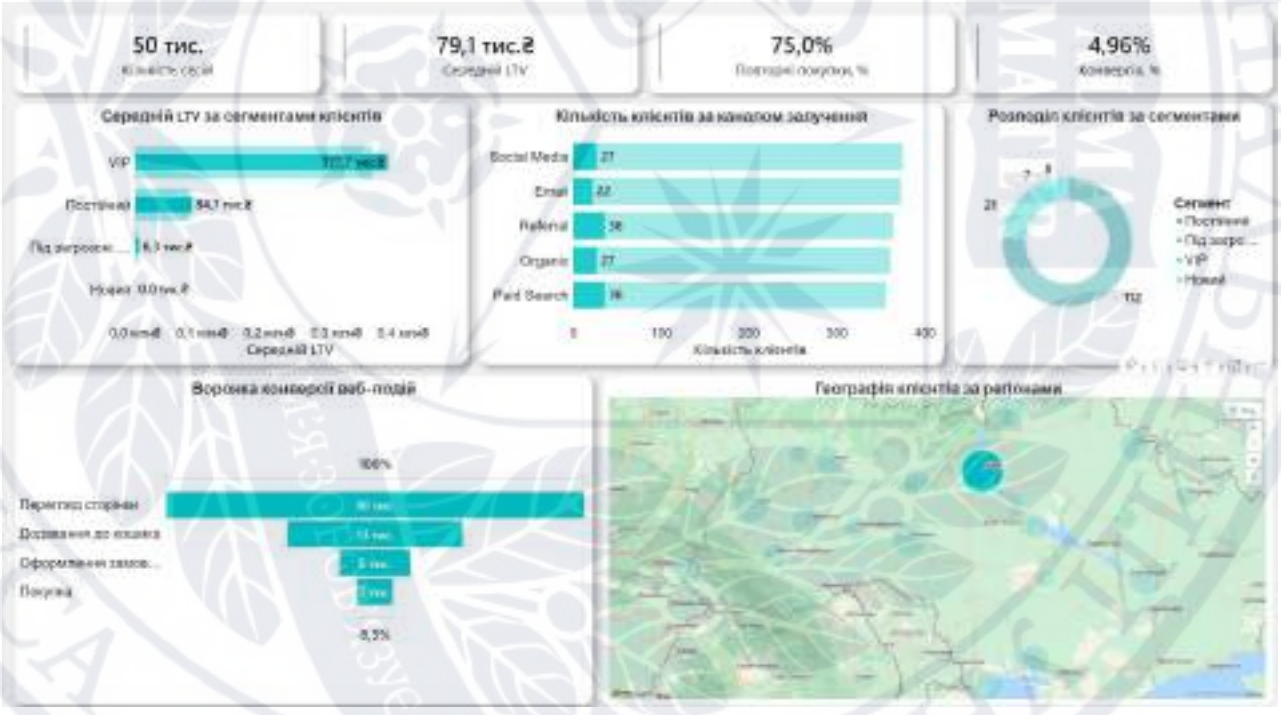

Рисунок А.4 – Крос-фільтрація за географічним розташуванням (місто Київ)

ДОДАТОК Б

DAG для завантаження бази даних у Google BigQuery

```
import os
from datetime import datetime

import pandas as pd
import pandas_gbq
import psycopg2
from airflow import DAG
from airflow.operators.python import PythonOperator

TABLES = [
    "countries",
    "regions",
    "cities",
    "person",
    "person_contacts",
    "customers",
    "customer_addresses",
    "categories",
    "subcategories",
    "suppliers",
    "carriers",
    "products",
    "product_prices",
    "order_statuses",
    "channels",
    "payment_methods",
    "roles",
    "employees",
    "promotions",
    "promotion_products",
    "orders",
    "order_items",
    "order_status_history",
    "order_managers",
    "payments",
    "shipments",
    "reviews",
]


def upload_table(table_name: str) -> None:
    conn = psycopg2.connect(
        host=os.environ["POSTGRES_HOST"],
        port=os.environ["POSTGRES_PORT"],
        dbname=os.environ["POSTGRES_DB"],
        user=os.environ["POSTGRES_USER"],
        password=os.environ["POSTGRES_PASSWORD"],
    )
    df = pd.read_sql(f"SELECT * FROM {table_name}", conn)
```

```
    conn.close()
    pandas_gbq.to_gbq(
        df,
        destination_table=f"raw.{table_name}",
        project_id=os.environ["BQ_PROJECT_ID"],
        if_exists="replace",
    )


with DAG(
    dag_id="postgres_to_bq",
    schedule_interval=None,
    start_date=datetime(2026, 4, 1),
    catchup=False,
) as dag:
    tasks = {
        table: PythonOperator(
            task_id=f"upload_{table}",
            python_callable=upload_table,
            op_kwargs={"table_name": table},
        )
        for table in TABLES
    }

    tasks["countries"] >> tasks["regions"] >> tasks["cities"]
    tasks["countries"] >> tasks["carriers"]
    tasks["categories"] >> tasks["subcategories"]
    tasks["suppliers"]       >>       tasks["products"]       >>
tasks["product_prices"]
    tasks["roles"] >> tasks["employees"]
    tasks["person"] >> tasks["customers"]
    tasks["person"] >> tasks["employees"]
    tasks["person"] >> tasks["person_contacts"]
    tasks["customers"] >> tasks["customer_addresses"]
    tasks["promotions"] >> tasks["promotion_products"]
    [
        tasks["customers"],
        tasks["cities"],
        tasks["channels"],
        tasks["order_statuses"],
        tasks["promotions"],
    ] >> tasks["orders"]
    tasks["orders"] >> tasks["order_items"]
    tasks["order_items"] >> tasks["order_status_history"]
    tasks["orders"] >> tasks["order_managers"]
    tasks["orders"] >> tasks["payments"]
    tasks["orders"] >> tasks["shipments"]
    tasks["order_items"] >> tasks["reviews"]
```

ДОДАТОК В

Тригери та збережена процедура операційної бази даних PostgreSQL

```
CREATE OR REPLACE FUNCTION fn_decrease_stock()
RETURNS TRIGGER AS $$
BEGIN
    IF (SELECT stock_quantity FROM products WHERE id =
NEW.product_id) < NEW.quantity THEN
        RAISE EXCEPTION 'Insufficient stock for product id %',
NEW.product_id;
    END IF;

    UPDATE products
    SET stock_quantity = stock_quantity - NEW.quantity
    WHERE id = NEW.product_id;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER trg_decrease_stock
AFTER INSERT ON order_items
FOR EACH ROW
EXECUTE FUNCTION fn_decrease_stock();


CREATE OR REPLACE FUNCTION fn_log_status_change()
RETURNS TRIGGER AS $$
BEGIN
    IF OLD.status_id IS DISTINCT FROM NEW.status_id THEN
        INSERT INTO order_status_history (order_id, status_id,
changed_at)
        VALUES (NEW.id, NEW.status_id, NOW());
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER trg_log_status_change
AFTER UPDATE ON orders
FOR EACH ROW
EXECUTE FUNCTION fn_log_status_change();


CREATE TYPE order_item_input AS (
    product_id BIGINT,
    quantity   INTEGER
);

CREATE OR REPLACE PROCEDURE place_order(
    p_customer_id       BIGINT,
    p_address_id        BIGINT,
```

```
    p_channel_id        BIGINT,
    p_promotion_id      BIGINT,
    p_payment_method_id BIGINT,
    p_items             order_item_input[]
)
LANGUAGE plpgsql
AS $$
DECLARE
    v_order_id      BIGINT;
    v_status_new_id BIGINT;
    v_item          order_item_input;
    v_unit_price    NUMERIC(10, 2);
    v_stock         INTEGER;
    v_total         NUMERIC(10, 2) := 0;
    v_discount      NUMERIC(5, 2) := 0;
BEGIN
    SELECT id INTO v_status_new_id
    FROM order_statuses
    WHERE name = 'new';

    IF p_promotion_id IS NOT NULL THEN
        SELECT discount_percent INTO v_discount
        FROM promotions
        WHERE id = p_promotion_id
          AND valid_from <= CURRENT_DATE
          AND valid_to >= CURRENT_DATE;

        IF v_discount IS NULL THEN
            v_discount := 0;
        END IF;
    END IF;

    FOREACH v_item IN ARRAY p_items LOOP
        SELECT stock_quantity INTO v_stock
        FROM products
        WHERE id = v_item.product_id;

        IF v_stock IS NULL THEN
            RAISE EXCEPTION 'Товар з id % не існує',
v_item.product_id;
        END IF;

        IF v_stock < v_item.quantity THEN
            RAISE EXCEPTION 'Недостатньо товару з id % на складі',
v_item.product_id;
        END IF;
    END LOOP;

    INSERT INTO orders (customer_id, address_id, channel_id,
promotion_id, status_id)
```

```
    VALUES     (p_customer_id,     p_address_id,     p_channel_id,
p_promotion_id, v_status_new_id)
    RETURNING id INTO v_order_id;

    FOREACH v_item IN ARRAY p_items LOOP
        SELECT price INTO v_unit_price
        FROM product_prices
        WHERE product_id = v_item.product_id
          AND valid_from <= CURRENT_DATE
          AND (valid_to IS NULL OR valid_to >= CURRENT_DATE)
        ORDER BY valid_from DESC
        LIMIT 1;

        IF v_unit_price IS NULL THEN
            RAISE EXCEPTION 'Немає актуальної ціни для товару з id
%', v_item.product_id;
        END IF;

        INSERT INTO order_items (order_id, product_id, quantity,
unit_price)
        VALUES (v_order_id, v_item.product_id, v_item.quantity,
v_unit_price);

        v_total := v_total + (v_unit_price * v_item.quantity);
    END LOOP;

    IF v_discount > 0 THEN
        v_total := v_total * (1 - v_discount / 100);
    END IF;

    INSERT INTO payments (order_id, amount, payment_method_id,
status)
    VALUES (v_order_id, v_total, p_payment_method_id, 'pending');

    INSERT   INTO   order_status_history   (order_id,   status_id,
changed_at)
    VALUES (v_order_id, v_status_new_id, NOW());

    COMMIT;
END;
$$;
```

ДОДАТОК Д

SQL-моделі mart-шару аналітичного пайплайну dbt

```
-- fact_orders.sql
SELECT
  oi.id AS order_item_key,
  o.id AS order_id,
  o.customer_id AS customer_key,
  oi.product_id AS product_key,
  o.channel_id AS channel_key,
  o.promotion_id AS promotion_key,
  sh.carrier_id AS carrier_key,
  FORMAT_DATE('%Y%m%d', DATE(o.created_at)) AS date_key,
  oi.quantity,
  oi.unit_price,
  ROUND(oi.quantity * oi.unit_price, 2) AS line_total,
  ROUND(
    oi.quantity * oi.unit_price *
    COALESCE(pr.discount_percent, 0) / 100, 2
  ) AS discount_amount,
  ROUND(
    oi.quantity * oi.unit_price *
    (1 - COALESCE(pr.discount_percent, 0) / 100), 2
  ) AS net_amount,
  oi.is_returned,
  os.name AS order_status,
  pay.status AS payment_status,
  pay.amount AS payment_amount,
  CAST(
    TIMESTAMP_DIFF(sh.delivered_at, o.created_at, DAY)
    AS INT64) AS days_to_deliver,
  o.created_at AS order_created_at,
  sh.shipped_at,
  sh.delivered_at
FROM {{ ref('stg_order_items') }} oi
JOIN {{ ref('stg_orders') }} o ON o.id = oi.order_id
LEFT JOIN {{ ref('stg_order_statuses') }} os ON os.id = o.status_id
LEFT JOIN {{ ref('stg_payments') }} pay ON pay.order_id = o.id
LEFT JOIN {{ ref('stg_shipments') }} sh ON sh.order_id = o.id
LEFT JOIN {{ ref('stg_promotions') }} pr ON pr.id = o.promotion_id



-- fact_web_events.sql
SELECT
  we.session_id,
  we.customer_id AS customer_key,
  we.product_id AS product_key,
  FORMAT_DATE('%Y%m%d', DATE(we.event_timestamp)) AS date_key,
  we.event_type,
  we.page_url,
  we.device_type,
```

```
  we.event_timestamp
FROM {{ ref('stg_web_events') }} we



-- dim_date.sql
WITH date_spine AS (
  SELECT DATE_ADD(DATE('2023-01-01'), INTERVAL n DAY) AS date
  FROM UNNEST(GENERATE_ARRAY(0, 730)) AS n
)
SELECT
  FORMAT_DATE('%Y%m%d', date) AS date_key,
  date,
  EXTRACT(DAY FROM date) AS day,
  EXTRACT(ISOWEEK FROM date) AS week,
  EXTRACT(MONTH FROM date) AS month,
  CASE EXTRACT(MONTH FROM date)
  WHEN 1  THEN 'Січень'
    WHEN 2  THEN 'Лютий'
    WHEN 3  THEN 'Березень'
    WHEN 4  THEN 'Квітень'
    WHEN 5  THEN 'Травень'
    WHEN 6  THEN 'Червень'
    WHEN 7  THEN 'Липень'
    WHEN 8  THEN 'Серпень'
    WHEN 9  THEN 'Вересень'
    WHEN 10 THEN 'Жовтень'
    WHEN 11 THEN 'Листопад'
    WHEN 12 THEN 'Грудень'
  END AS month_name,
  EXTRACT(QUARTER FROM date) AS quarter,
  EXTRACT(YEAR FROM date) AS year,
  IF(EXTRACT(DAYOFWEEK FROM date) IN (1, 7), TRUE, FALSE) AS
is_weekend
FROM date_spine



-- dim_customers.sql
SELECT
  c.id AS customer_key,
  p.first_name,
  p.last_name,
  p.first_name || ' ' || p.last_name AS full_name,
  pc.email,
  pc.phone,
  ca.street,
  ca.house_number,
  ca.postal_code,
  ci.name AS city,
  r.name AS region,
  cs.segment,
```

```
  cs.lifetime_value,
  cs.acquisition_channel,
  cs.first_purchase_date,
  c.created_at
FROM {{ ref('stg_customers') }} c
LEFT JOIN {{ ref('stg_person') }} p ON p.id = c.person_id
LEFT JOIN {{ ref('stg_person_contacts') }} pc ON pc.person_id =
c.person_id
LEFT JOIN {{ ref('stg_customer_addresses') }} ca ON ca.customer_id
= c.id AND ca.is_default = TRUE
LEFT JOIN {{ ref('stg_cities') }} ci ON ci.id = ca.city_id
LEFT JOIN {{ ref('stg_regions') }} r ON r.id = ci.region_id
LEFT JOIN {{ ref('stg_customer_segments') }} cs ON cs.customer_id =
c.id


-- dim_products.sql
SELECT
  p.id AS product_key,
  p.name,
  p.sku,
  sub.name AS subcategory,
  cat.name AS category,
  sup.name AS supplier,
  pp.price AS current_price,
  sp.purchase_price,
  ROUND((pp.price - sp.purchase_price) / pp.price * 100, 2) AS
margin_percent
FROM {{ ref('stg_products') }} p
LEFT JOIN {{ ref('stg_subcategories') }} sub ON sub.id =
p.subcategory_id
LEFT JOIN {{ ref('stg_categories') }} cat ON cat.id = sub.category_id
LEFT JOIN {{ ref('stg_suppliers') }} sup ON sup.id = p.supplier_id
LEFT JOIN {{ ref('stg_product_prices') }} pp
  ON pp.product_id = p.id AND pp.valid_to IS NULL
LEFT JOIN {{ ref('stg_supplier_prices') }} sp ON sp.product_sku =
p.sku
```

ДОДАТОК Е

Частина SQL-моделей staging-шару аналітичного пайплайну dbt

```
-- stg_orders.sql
SELECT
    id,
    customer_id,
    address_id,
    channel_id,
    promotion_id,
    status_id,
    TIMESTAMP(created_at) AS created_at
FROM {{ source('raw', 'orders') }}


-- stg_order_items.sql
SELECT
    id,
    order_id,
    product_id,
    quantity,
    CAST(unit_price AS NUMERIC) AS unit_price,
    is_returned,
    return_reason
FROM {{ source('raw', 'order_items') }}


-- stg_web_events.sql
SELECT
    session_id,
    customer_id,
    event_type,
    page_url,
    product_id,
    TIMESTAMP(`timestamp`) AS event_timestamp,
    device_type
FROM {{ source('raw', 'web_events') }}


-- macros/generate_schema_name.sql
{% macro generate_schema_name(custom_schema_name, node) -%}
  {%- if custom_schema_name is none -%}
    {{ target.schema }}
  {%- else -%}
    {{ custom_schema_name | trim }}
  {%- endif -%}
{%- endmacro %}
```

ДОДАТОК Ж

Скрипти генерації зовнішніх джерел даних

```
# generate_csv.py
import csv
import os
import random
from collections import Counter
from pathlib import Path

import psycopg2
from dotenv import load_dotenv

load_dotenv()

DB_CONFIG = {
    'host': os.getenv('DB_HOST', 'localhost'),
    'port': int(os.getenv('DB_PORT', 5432)),
    'dbname': os.getenv('DB_NAME', 'ecommerce_db'),
    'user': os.getenv('DB_USER', 'postgres'),
    'password': os.getenv('DB_PASSWORD', ''),
}

random.seed(42)

CHANNELS = ['Organic', 'Paid Search', 'Social Media', 'Referral',
'Email']


def segment_by_ltv(ltv):
    if ltv == 0:
        return 'New'
    if ltv < 10_000:
        return 'At-risk'
    if ltv < 250_000:
        return 'Regular'
    return 'VIP'


def main():
    conn = psycopg2.connect(**DB_CONFIG)
    cur = conn.cursor()

    cur.execute("""
        SELECT
            c.id,
            COALESCE(SUM(p.amount)   FILTER   (WHERE   p.status   =
'completed'), 0) AS lifetime_value,
            MIN(o.created_at)::date AS first_purchase_date
        FROM customers c
```

```
        LEFT JOIN orders o ON o.customer_id = c.id
        LEFT JOIN payments p ON p.order_id = o.id
        GROUP BY c.id
        ORDER BY c.id
    """)
    rows = cur.fetchall()
    cur.close()
    conn.close()

    output_path = Path(__file__).resolve().parents[2] / 'data_sources' / 'csv' / 'customer_segments.csv'
    segments = []

    with open(output_path, 'w', newline='', encoding='utf-8') as f:
        writer = csv.writer(f)
        writer.writerow(['customer_id', 'segment', 'lifetime_value', 'acquisition_channel', 'first_purchase_date'])
        for customer_id, lifetime_value, first_purchase_date in rows:
            ltv = round(float(lifetime_value), 2)
            seg = segment_by_ltv(ltv)
            segments.append(seg)
            writer.writerow([
                customer_id,
                seg,
                ltv,
                random.choice(CHANNELS),
                first_purchase_date if first_purchase_date is not None else '',
            ])

    print(f'Saved {len(rows)} rows -> {output_path}')
    for seg, count in sorted(Counter(segments).items()):
        print(f'  {seg}: {count}')


main()


# generate_json.py
import json
import os
import random
import uuid
from collections import Counter
from datetime import datetime, timedelta
from pathlib import Path

import psycopg2
from dotenv import load_dotenv
```

```
load_dotenv()

DB_CONFIG = {
    'host': os.getenv('DB_HOST', 'localhost'),
    'port': int(os.getenv('DB_PORT', 5432)),
    'dbname': os.getenv('DB_NAME', 'ecommerce_db'),
    'user': os.getenv('DB_USER', 'postgres'),
    'password': os.getenv('DB_PASSWORD', ''),
}

random.seed(42)

TOTAL_EVENTS = 50_000
EVENT_TYPES = ['page_view', 'add_to_cart', 'checkout', 'purchase']
EVENT_WEIGHTS = [60, 25, 10, 5]
DEVICE_TYPES = ['desktop', 'mobile', 'tablet']
DEVICE_WEIGHTS = [50, 40, 10]
STATIC_URLS = ['/', '/catalog', '/about']

DATE_START = datetime(2023, 1, 1)
DATE_RANGE_SEC  =  int((datetime(2024,  12,  31,  23,  59,  59)  -
DATE_START).total_seconds())


def random_ts():
    return   (DATE_START   +   timedelta(seconds=random.randint(0,
DATE_RANGE_SEC))).strftime('%Y-%m-%dT%H:%M:%S')


def main():
    conn = psycopg2.connect(**DB_CONFIG)
    cur = conn.cursor()

    cur.execute("SELECT id FROM customers ORDER BY id")
    customer_ids = [r[0] for r in cur.fetchall()]

    cur.execute("SELECT id, sku FROM products ORDER BY id")
    products = cur.fetchall()

    cur.execute("SELECT name FROM categories ORDER BY id")
    category_urls = [f'/category/{r[0]}' for r in cur.fetchall()]

    cur.close()
    conn.close()

    page_view_urls = STATIC_URLS + category_urls

    events = []
    for _ in range(TOTAL_EVENTS):
```

```
        event_type = random.choices(EVENT_TYPES,
weights=EVENT_WEIGHTS, k=1)[0]
        customer_id = None if random.random() < 0.30 else
random.choice(customer_ids)

        if event_type == 'page_view':
            page_url = random.choice(page_view_urls)
            product_id = None
        elif event_type == 'checkout':
            page_url = f'/product/{random.choice(products)[1]}'
            product_id = None
        else:
            prod = random.choice(products)
            page_url = f'/product/{prod[1]}'
            product_id = prod[0]

        events.append({
            'session_id': str(uuid.uuid4()),
            'customer_id': customer_id,
            'event_type': event_type,
            'page_url': page_url,
            'product_id': product_id,
            'timestamp': random_ts(),
            'device_type': random.choices(DEVICE_TYPES,
weights=DEVICE_WEIGHTS, k=1)[0],
        })

    output_path = Path(__file__).resolve().parents[2] /
'data_sources' / 'json' / 'web_events.json'

    with open(output_path, 'w', encoding='utf-8') as f:
        json.dump(events, f, ensure_ascii=False, indent=2)

    print(f'Saved {len(events)} events -> {output_path}')
    for et, count in sorted(Counter(e['event_type'] for e in
events).items()):
        print(f'  {et}: {count}')


main()
```

ДОДАТОК К

Порівняльний аналіз класичних OLAP-систем та хмарних аналітичних сховищ

Таблиця К.1

Критерій	**Класичні OLAP-системи**	**Хмарні аналітичні сховища (CDW)**
Архітектура зберігання	Рядково-орієнтована або кубічна; дані попередньо агрегуються у OLAP-куби	Колонково-орієнтована; читаються лише потрібні стовпці без попередньої агрегації
Масштабування	Вертикальне; обмежено апаратними можливостями одного вузла	Горизонтальне та еластичне; обчислення і зберігання масштабуються незалежно
Підготовка даних	Потребує побудови кубів; зміна схеми – трудомістка операція	Трансформація SQL безпосередньо у сховищі (підхід ELT)
Вартість інфраструктури	Висока: власне обладнання, ліцензії, команда адміністраторів	Оплата за фактичне споживання; без капітальних витрат
Підтримка різнорідних джерел	Обмежена; орієнтована на структуровані реляційні джерела	Нативна підтримка структурованих, JSON, Parquet та зовнішніх джерел
Типові представники	MS SQL Server Analysis Services, Oracle Essbase, IBM Cognos TM1	Google BigQuery, Amazon Redshift

ДОДАТОК Л

Порівняльний аналіз інструментів візуалізації даних

Таблиця Л.1

Критерій	**Power BI**	**Tableau**	**Looker Studio**	**Metabase**
Розробник	Microsoft	Salesforce	Google	Metabase, Inc.
Ліцензування	Desktop; Pro ~10 дол./міс.	Creator ~75 дол./міс.	Безкоштовно	Self-hosted; Cloud ~500 дол./міс.
Складні обчислення	DAX – повна підтримка	LOD-вирази, Table Calc	Обмежена	Відсутня (лише SQL)
Інтеграція з BigQuery	Нативний конектор	Є, потребує налаштування	Нативна (Google-екосистема)	Через SQL-підключення
Семантична модель	Повна (таблиці, зв'язки, міри)	Є (Data Source)	Обмежена	Відсутня
Цільова аудиторія	Бізнес-аналітики, Microsoft-середовища	Data analysts, enterprise	Marketing, Google-екосистема	Розробники, DB-орієнтовані команди

ДОДАТОК М

Порівняльні характеристики інструментів аналітичного пайплайну

Таблиця М.1

Критерій	**Airflow**	**BigQuery**	**dbt**	**Power BI**
Призначення	Оркестрація задач	Зберігання і виконання SQL	Декларативна трансформація	Візуалізація і DAX-обчислення
Роль у ELT	Extract + Load (E і L)	Сховище для всіх шарів	Transform (T)	Кінцева аналітика
Мова конфігурації	Python (DAG)	SQL + JSON API	SQL + Jinja	DAX, M (Power Query)
Підтримка залежностей	Явний граф задач (DAG)	Немає (виконує запити)	Автоматичний граф ref()	Немає (семантична модель)
Тестування якості даних	Немає (лише моніторинг задач)	Немає нативного	Вбудовані тести (not_null, unique)	Немає
Розгортання	Docker Compose (LocalExecutor)	Google Cloud (serverless)	CLI або dbt Cloud	Desktop або Power BI Service