НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

УДК 004-027.87:336.71

«ПОГОДЖЕНО»	**«ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ»**
Декан факультету інформаційних технологій	Завідувач кафедри комп'ютерних наук
Глазунова О.Г., д.п.н., професор	Голуб Б.Л., к.т.н., доцент
_____	_____
____ _____ 2023 р.	**____ _____ 2023 р.**

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему Аналітична система дистанційного управління діяльністю фінансової установи

Спеціальність 122 Комп'ютерні науки

_____
(код і назва)

Освітня програма Інформаційні управляючі системи та технології

(назва)

Орієнтація освітньої програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Гарант освітньої програми

кандидат технічних наук, доцент _____ Голуб Белла Львівна
(науковий ступінь та вчене звання) (підпис) (ПІБ)

Керівник магістерської кваліфікаційної роботи

к.фіз.-м.н., доцент _____ Кириченко В.В.
(науковий ступінь та вчене звання) (підпис) (ПІБ)

Виконав _____ Гордій Я. В.
(підпис) (ПІБ студента)

КИЇВ-2023

НАЦІОНАЛЬНИЙ УНІВЕРСТИТЕТ БІОРЕСУРСІВ
І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

ЗАТВЕРДЖУЮ
Завідувач кафедри
комп'ютерних наук
(назва кафедри)
к.тех.н., доцент Голуб Б.Л.
(вчене звання і ступінь) (підпис) (ініціали і прізвище)

«_____» _____ 2023 р.

З А В Д А Н Н Я

ДО ВИКОНАННЯ МАГІСТЕРСЬКОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ СТУДЕНТУ

Гордія Ярослава Вікторовича
(прізвище, ім'я, по батькові)

Спеціальність 122 Компютерні науки
(код і назва)

Освітня програма «Інформаційні управляючі системи та технології»
(назва)

Орієнтація освітньої програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Тема магістерської кваліфікаційної роботи: Аналітична система дистанційного управління діяльністю фінансової установи
затверджена наказом ректора НУБіП України від " 30 " грудня 2023р. № 1940-С

Термін подання завершеної роботи на кафедру 5 листопада 2023 року
(рік, місяць, число)

Вихідні дані до магістерської кваліфікаційної роботи для цієї роботі дані були згенеровані з відповідністю до ринку впровадження підсистеми у компанії, а саме: партнери, типи активностей, користувачі, пристрої користувачів, сесії користувачів у системі з часом

Перелік питань, що підлягають дослідженню:

1. Спроектувати підсистему онлайн установи, яка отримує дані про діяльність користувачів та обробляє їх з подальшим аналізом
2. Реалізувати автоматичну перевірку якості вхідних даних на етапі вивантаження
3. Виправити та дослідити вплив цих даних
4. Дослідити користувацьку активність та вивести бізнес-метрики для стратегічного прийняття рішень

Перелік графічного матеріалу (за потреби) _____
_____

Дата видачі завдання "15'' грудня 2022 р.

Керівник магістерської кваліфікаційної роботи _____ Кириченко В.В.
(підпис) (прізвище та ініціали)

Завдання прийняв до виконання _____ Гордій Я.В
(підпис) (прізвище та ініціали студента)

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

1. Рис. – рисунок
2. Табл. – таблиця
3. СЕД – система електронного документообігу.
4. СКП – Система комп’ютерного прогнозування
5. BI - Business Intelligence
6. ER – Entity Relation
7. КП – комп’ютерне прогнозування
8. СУБД – система управління базою даних
9. HOLAP – Hybrid OLAP.
10. СД – сховище даних
11. MOLAP – Multidimensional OLAP
12. OLAP – On-Line Analytical Processing.
13. ROLAP – Relational OLAP
14. SSAS – SQL Server Analysis Services.
15. SQL – Structured Query Language
16. API – Application Programming Interface
17. CTR - Click-Through Rate
18. HTTP – Hyper Text Transfer Protocol
19. IDE – Integrated development environment
20. HTML – Hyper Text Markup Language
21. DBMS - Database Management System
22. ETL – Extract, Transform, Load
23. DAG - Directed Acyclic Graphs
24. GDPR - General Data Protection Regulation
25. FE – FrontEnd
26. AWS - Amazon Web Services

ВСТУП

1. СИСТЕМНИЙ АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Опис предметної області

Фінансові установи є ключовими гравцями в економіці країни і, як такі, мають важливу роль у забезпеченні стабільності фінансової системи. Для досягнення цієї мети, фінансові установи повинні вести свою діяльність ефективно та ефективно управляти своїми ресурсами.

Одним з ключових аспектів ефективного управління є використання аналітичних систем, які надають важливу інформацію для прийняття рішень. Аналітична система дистанційного управління діяльністю фінансової установи є однією з таких систем, що дозволяє відстежувати ключові показники діяльності фінансової установи, проводити аналіз даних та приймати рішення на основі цих даних.

Однак, успіх такої системи залежить від якості даних, які вона використовує. Якісні дані є ключовим елементом будь-якої аналітичної системи, оскільки вони дозволяють забезпечити точність та достовірність аналізу. Тому, для досягнення максимального ефекту від використання аналітичної системи дистанційного управління діяльністю фінансової установи, необхідно забезпечити якість даних.

У цій роботі буде досліджено підходи та методи, що дозволять ефективно здійснювати аналіз даних, буди обізнанаными щодо якості даних та результуючий аналіз з прогнозуванням у фінансовій установі. Крім того, робота буде зосереджена на визначенні факторів, що впливають на якість даних, та розробці рекомендацій з їх поліпшення. Оскільки якість даних є ключовим

елементом для успішної роботи аналітичної системи дистанційного управління діяльністю фінансової установи та її користувачами.

Актуальність теми полягає в тому, що фінансові установи все більше використовують аналітичні системи для прийняття рішень та управління діяльністю. Однак, забезпечення якості даних, які використовуються в таких системах, залишається найбільш важливим завданням.

Недостатня якість даних може призвести до неточностей та недостовірної інформації, що може негативно вплинути на прийняття рішень та діяльність фінансової установи в цілому. Окрім того, зростаючі обсяги даних та швидкість їх збірного ставлять перед фінансовими установами виклик із забезпеченням якості та ефективності аналітичних систем.

1.2 Визначення вимог

Сама установа являє собою вебплатформу, яка допомагає її користувачам з вибором пропозицій інших фінансових установ, приклад пропозицій: вигідні кредитні умови, дострокові погашення боргів, студентські займи та ін. Аналітична система самої установи націлена на аналіз даних для подальшої співпраці між самою установою та її клієнтами, оскільки прибуток сама установа має за фіксовані дивіденди за перегляди, кліки, переходи клієнтів.

Вимоги до аналітичної системи дистанційного управління діяльністю фінансової установи, яка є веб продуктом для сприяння вибору фінансових пропозицій, поділяються на дві основні категорії:

- Оперативна діяльність і збір даних
- Аналітична та прогностична діяльність

Розглянемо кожну категорію докладніше.

- Оперативна діяльність і збір даних

Основна задача цієї частини - забезпечити надійний і оперативний збір даних користувачів. Сюди входять:

- Збір даних користувачів

Тут відбувається збір всієї активності користувачів: перегляди, кліки, переходи, вибрані фінансові пропозиції, тощо. Система повинна мати змогу фіксувати ці дані в реальному часі.

- Оперативна база даних

Це єдина точка зберігання первинних даних, з якої вони можуть бути використані для подальшого аналізу або агрегації.

- Обробка і агрегація даних

Всі зібрані дані підлягають агрегації та перевірці на аномалії та відхилення.

- Аналітична та прогностична діяльність

Ця частина фокусується на глибокому аналізі та прогнозуванні на основі зібраних даних. Сюди входять:

- Сховище даних

Всі агреговані та оброблені дані зберігаються тут для подальшого аналізу.

- Візуалізація та аналіз даних

Засоби для візуалізації ключових показників ефективності, таких як CTR, кореляція, тощо. Це допомагає у взаємодії з клієнтами та оптимізації доходів.

- Прогнозування та оптимізація

Механізми для прогнозування активності користувачів та потенційних доходів.

- Сповіщення та алерти

Система для моніторингу ключових показників та автоматичної відправки повідомлень у разі виявлення аномалій або відхилень.

Оперативна діяльність і збір даних – це фундаментальна частина системи. Це дозволяє налаштувати потоки даних та забезпечити їх якість. Система повинна взаємодіяти з користувачами, фіксувати їхню активність та зберігати ці дані для подальшого аналізу.

Аналітична та прогностична діяльність – це важливий інструмент для оптимізації прибутку установи. Це включає в себе не лише аналіз минулої активності, але і прогнозування майбутніх трендів на основі зібраних даних.

Аналітична та прогностична діяльність є важливим інструментом для оптимізації прибутку установи. Ця частина не лише дозволяє аналізувати минулу активність, але і прогнозувати майбутні тренди на основі зібраних даних. Згідно з дослідженням McKinsey & Company, компанії, які використовують дані для прийняття рішень, на 23% ефективніші в приваблені нових клієнтів і на 6% ефективніші в збереженні існуючих [1].

Враховуючи специфіку діяльності фінансової установи, ці вимоги є критично важливими для успішної реалізації проекту.

1.3 Аналіз вимог до програмної та апаратної системи

Після розгляду основних вимог до функціональності системи можна перейти до аналізу вимог до програмного та апаратного забезпечення. Специфічність цієї магістерської роботи полягає в комплексному підході до аналізу діяльності фінансової установи, використовуючи різні технологічні рішення.

Вимоги до клієнтської частини:

Браузер: Важливо мати браузер з підтримкою нових версій JavaScript..

Оперативна пам'ять: Мінімум 4 ГБ оперативної пам'яті буде достатньо для нормального функціонування сайту.

Вимоги до серверної частини:

Airflow: Використовується для оркестрації робочих процесів ETL (Extract, Transform, Load).

SQL Server: Використовується як основна OLTP (Online Transaction Processing) та OLAP (Online Analytical Processing) системи баз даних.

Python: Використовується для візуалізації та прогнозування даних.

Щодо збору даних, хоча сторонній сервіс Segment часто використовується для цієї цілі, в даному контексті буде реалізована генерація даних наближених до реальних, через конфіденційність користувачів.

Додаткові інструменти:

Docker: Для контейнеризації та спрощення розгортання.

Telegram API: Використовується для нотифікації аномалій в даних.

Ці вимоги виходять з припущення, що система буде реалізована на основі клієнт-серверної архітектури. Кожний з цих інструментів був обраний з огляду на його специфічні переваги та можливості, які будуть детальніше розглянуті в наступних розділах.

1.4 Аналіз наявних рішень

Перед розробкою аналітичної системи дистанційного управління діяльністю фінансової установи важливо провести аналіз наявних рішень на ринку. Це допоможе зрозуміти, які функції вже реалізовані, і які є ще ноу-хау для індустрії.

- Системи Бізнес Інтелекту (BI):

Tableau: Сильний інструмент для візуалізації даних, але може бути дорогим для малого та середнього бізнесу.

Power BI: Вбудований в екосистему Microsoft, легко інтегрується з іншими продуктами Microsoft, але може бути обмежуючим для користувачів інших платформ.

- Системи збору даних:

Google Analytics: Безкоштовний та широко використовуваний, але має обмеження по конфіденційності даних.

Segment: Дозволяє легко інтегрувати декілька джерел даних, але в даній роботі не використовується через проблеми з конфіденційністю.

- Системи управління базами даних (DBMS):

SQL Server: Часто використовується в корпоративних системах, підтримує як OLTP, так і OLAP.

PostgreSQL: Відкритий код, висока надійність, але може бути менш продуктивним для деяких задач OLAP.

- Оркестрація та моніторинг:

Apache Airflow: Широко використовується для оркестрації ETL-процесів, але має високий поріг входу.

Kubernetes: Для оркестрації контейнерів, але може бути занадто складним для невеликих проектів.

Кожна з цих систем має свої переваги та недоліки, і вибір конкретного рішення повинен базуватися на специфічних вимогах проекту. В нашому випадку, система буде використовувати комбінацію SQL Server для бази даних, Python та Power BI для аналітики та прогнозування, та Airflow для оркестрації, з додатковою увагою до конфіденційності даних.

Комплексні рішення в аналітичних системах для фінансових установ:

- Salesforce Financial Services Cloud

Salesforce працює на основі концепції Software as a Service (SaaS), що забезпечує гнучкість та масштабованість. Salesforce було засновано в 1999 році і з самого початку зосереджувалося на хмарних рішеннях. Їх фінансовий продукт створено спеціально для фінансових установ і запущено в 2016 році.

- Характеристики: Це комплексний продукт, що об'єднує CRM, аналітику даних та інтеграцію з різними фінансовими системами.
- Переваги: Висока масштабованість, налаштування під конкретні бізнес-потреби.
- Недоліки: Висока вартість, складність впровадження.

- Oracle Financial Services Analytical Applications (OFSAA)

  OFSAA базується на концепції Enterprise Risk Management (ERM), що допомагає фінансовим установам оцінювати та керувати ризиками. Oracle було засновано в 1977 році і є одним з піонерів в області баз даних. OFSAA було введено на ринок в 2008 році для специфічних потреб фінансового сектору.

  - Характеристики: Продукт фокусується на рішеннях для ризик-менеджменту, звітності, та відповідністю в фінансовому секторі.
  - Переваги: Висока надійність, широкий функціонал.
  - Недоліки: Висока вартість, можлива складність в інтеграції з іншими системами.

- SAP for Banking

  Продукт розроблено на основі концепції Business Process Management (BPM), що допомагає оптимізувати банківські процеси та забезпечує інтеграцію з ERP системами. SAP було засновано в 1972 році в Німеччині. Продукт для банківського сектору є частиною їх широкої лінійки продуктів і було введено в експлуатацію в середині 1990-х.

  - Характеристики: Інтеграція з ERP системами, фокус на оптимізації банківських операцій.
  - Переваги: Масштабованість, глобальна підтримка.
  - Недоліки: Висока вартість та складність впровадження.

Спільною особливістю цих рішень є їхня орієнтація на великі фінансові установи і спрямованість на комплексне вирішення проблем. Наприклад, Forrester Research підкреслює, що комплексні рішення можуть забезпечити глобальний підхід до управління ризиками, дозволяючи установам краще розуміти та оптимізувати свою діяльність. [2] З їх дослідженням складно не

погодитись, однак для малих бізнесів такі рішення будуть занадто дорогими та навряд чи окупляться в короткий чи, навіть, середньостроковий термін часу. До того ж, оскільки ці рішення є на ринку вже досить тривалий час вони не будуть гнучкими в плані функціональності, тому система, яка створена в рамках цього дипломного проєкту, має перевагу саме в гнучкості функціональності та меншій вартості.

2. МОДЕЛЮВАННЯ СИСТЕМИ

2.1 Джерело надходження даних

Ця платформа є одним з ключових ресурсів для індивідів, які прагнуть краще розуміти свої фінансові можливості та оптимізувати фінансове планування. Вона надає комплексний набір інструментів, статей, рекомендацій та аналітичних звітів, які допомагають користувачам приймати обґрунтовані фінансові рішення.

Основні Функції:

- Порівняння фінансових продуктів: Платформа дозволяє користувачам порівнювати різні фінансові продукти, такі як кредитні картки, позики, страхування, і так далі.
- Освітній Контент: Надається велика кількість статей, відео та інших освітніх матеріалів, які варіюються від базових порад до глибоких аналітичних досліджень.
- Фінансові інструменти: Користувачі можуть скористатися різними калькуляторами, планувальниками бюджету та іншими інструментами для практичного аналізу своїх фінансів.

- Персоналізація: На основі даних користувача, платформа може рекомендувати специфічні фінансові продукти або стратегії, що найкраще підходять для їх потреб.

Завдяки цим функціям, платформа стає централізованим ресурсом для всіх, хто прагне оптимізувати свої фінанси і приймати обґрунтовані рішення.

В сучасному фінансовому ландшафті з'являється все більше індивідуальних потреб та цілей. Розуміння типів користувачів є ключовим для ефективної адаптації сервісів, їх функціональності та контенту. Нижче наведено кілька основних типів користувачів, які часто взаємодіють з фінансовими платформами.

Типи користувачів:

- Новачки у фінансах: Люди, які тільки починають орієнтуватися в фінансовому світі. Їм потрібні базові поради та рекомендації.
- Досвідчені інвестори: Користувачі, які вже мають певний фінансовий досвід і шукають специфічну інформацію або інструменти для аналізу.
- Пошукачі кредитів: Люди, які шукають кредитні картки, іпотеку або особисті позики.
- Студенти: Молоді люди, які шукають інформацію про студентські позики, стипендії, тощо.
- Батьки: Люди, які планують або уже мають дітей і шукають фінансові рішення з огляду на це (наприклад, створення накопичувального фонду).

Для зображеня того як саме користувачі генерують дані своїми діями допоможе діаграма User Flow. User Flow (користувацький потік) — це візуальна схема, яка показує послідовність кроків, які користувач виконує, щоб досягти

певної мети на веб-сайті або в мобільній аплікації. Це ключовий елемент в процесі розробки користувацького інтерфейсу (UI) і користувацького досвіду (UX), і він дає можливість краще зрозуміти, як користувачі взаємодіють з продуктом. Один з основних джерел, які займаються вивченням теми User Flow, — це книга "Don't Make Me Think" від Стіва Круга. Вона розглядає принципи створення інтуїтивно зрозумілих веб-сайтів і аплікацій, і одним з її ключових пунктів є значимість розуміння того, як користувачі переміщуються по платформі.[3] В реальному світі, User Flow часто представляється у вигляді діаграм, які можуть бути створені в ручну або за допомогою спеціалізованого ПЗ, такого як Adobe XD, Sketch або Figma. Схеми включають в себе різні етапи взаємодії, починаючи з входу на платформу і закінчуючи досягненням конечної мети, такої як покупка товару, реєстрація, або отримання інформації.

Особливу увагу при розробці User Flow слід приділити "болючим точкам" — моментам, де користувачі можуть зазнати труднощів або фрустрацій. Розуміння та оптимізація цих аспектів може значно покращити загальний користувацький досвід.

User Flow — не лише інструмент для дизайнерів, але і для стратегів та маркетологів. Він може бути використаний для A/B тестування, для вивчення поведінки користувачів, а також для планування маркетингових кампаній. Інтеграція User Flow в загальний процес розробки може допомогти команді сфокусуватися на найважливіших аспектах продукту з точки зору користувача.

З урахуванням цих принципів, User Flow стає не просто схемою, а стратегічним інструментом, який взаємодіє з різними аспектами бізнесу та розробки.

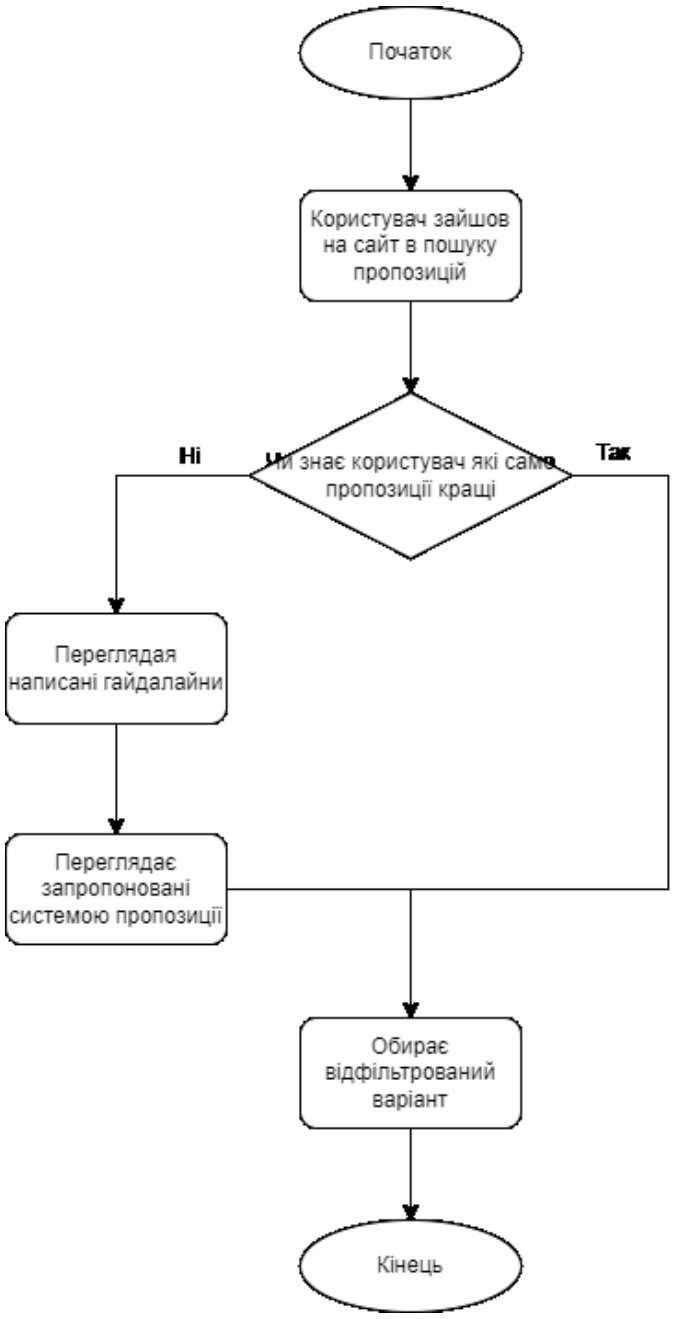


Рис. 1 User Flow діаграма

На рис.1 зображено високорівневу діаграму того, як користувачі використовують систему, для подальшого аналізу цієї інформації у даному проєкті.

Сама дипломна робота сфокусована на підсистемі яка проводить збір, модифікацію та аналіз даних, заснованих на діях користувачів у перегляді, кліках та переходах на пропозиції від фінансових партнерів платформи.

2.2 Діаграма прецедентів

Використовуючи UML (Unified Modeling Language), можна здійснити комплексний аналіз та візуалізацію предметної області. Це не тільки дозволяє глибше зрозуміти структуру та динаміку галузі, але і виявляє можливості для її оптимізації та масштабування. Засоби UML допомагають якісно аналізувати зв'язки між сутностями, функціональні ролі учасників, а також інтеграцію з іншими інформаційними системами. Використовується для аналізу та проектування систем, включаючи фінансові. Діаграми прецедентів, класів, активності та інші допомагають у візуалізації та аналізі системи [4].

Що стосується діаграми прецедентів, вона є ключовим інструментом для візуалізації взаємодій між акторами та випадками використання (прецедентами). Діаграма не тільки об’єктивізує процес збору вимог, але і служить мостом для ефективної комунікації між розробниками, аналітиками та кінцевими користувачами. Вона допомагає в ідентифікації функціональних аспектів системи, виявленні потенційних ризиків, а також у плануванні розробки та тестування.[5] В UML діаграма прецедентів може узагальнити деталі користувачів вашої системи (також відомих як актори) та їх взаємодію з системою. Щоб створити його, ви будете використовувати набір спеціалізованих символів і з’єднувачів. Використання діаграми прецедентів може допомогти команді обговорити та представити:

- Сценарії, у яких ваша система або програма взаємодіє з людьми, організаціями або зовнішніми системами
- Цілі, яких ваша система або програма допомагає цим об’єктам (відомим як актори) досягти
- Область вашої системи

Основні елементи діаграми - учасник та прецедент

Учасник - це безліч логічно пов'язаних ролей, що виконуються при взаємодії з прецедентами чи сутностями (система, підсистема чи клас). Учасником може бути людина чи інша система, підсистема чи клас, які є чимось поза сутністю. Графічно учасник зображується "чоловічком".

Прецедент (use case) - опис безлічі послідовних подій, що виконуються системою, які призводять до результату, що спостерігається учасником. Прецедент представляє поведінку сутності, описуючи взаємодію між учасниками та системою. Прецедент не показує, як досягається деякий результат, а тільки що саме виконується. Прецеденти позначаються дуже простим чином - у вигляді еліпса, всередині якого вказано його назву.

В діаграмі прецедентів (рис.1), яка описує дану систему, використані наступні актори:

- Користувач – кінцевий користувач системи який виконує діяльність на сайті
- Платформа даних клієнтів – система яка збирає дані активності користувача, агрегує та записує
- Дата інженер – підтримує діяльність Платформи даних клієнтів, виконує додаткове агрегування та контролює якість даних
- Аналітик – аналізує отримані дані

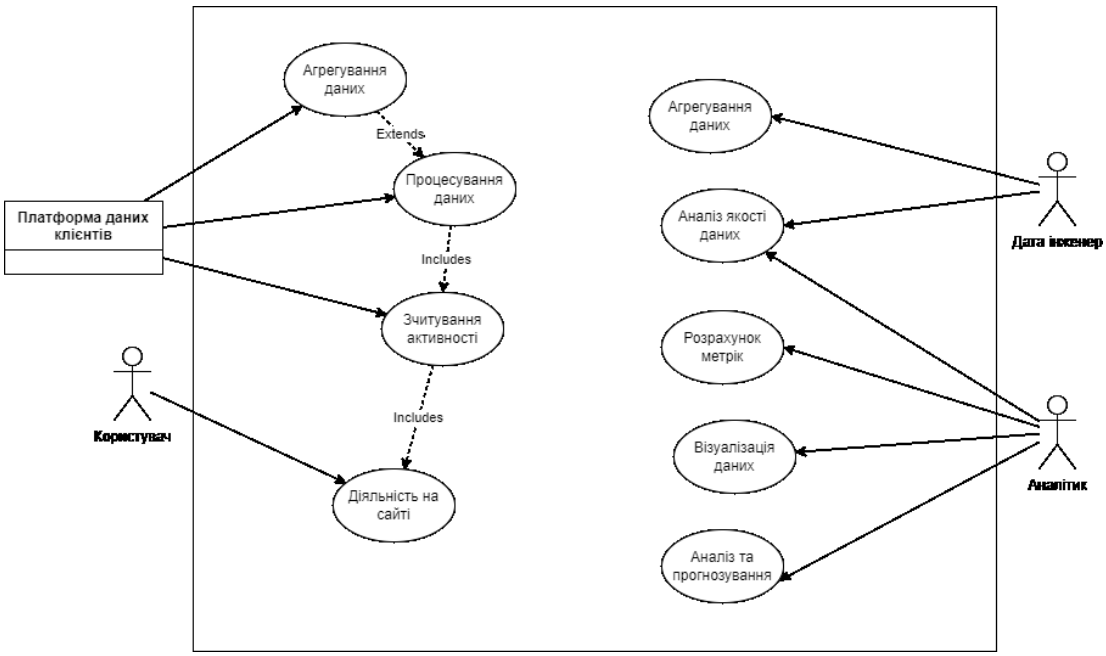


Рис. 2 Діаграмма прецедентів

Назва прецеденту: Аналіз даних діяльності користувачів

Мета прецеденту: Аналіз та прогноз даних діяльності користувачів, направлену на отримання найбільшої вигоди

Сценарії:

- Оптимістичний:
  - Користувач виконує якусь діяльність на сайті
  - Платформа даних клієнтів зчитує та процесує дані
  - Дані процесовані та вивантажені
  - Дата інженер агрегує дані та проводить перевірку даних
  - Аналітик робить аналіз та прогноз з розробкою звітів
- Прагматичний
  - Умова 1: команда реалізувала платформу даних клієнтів некоректно
  - Умова 2: девайс користувача некоректно передає інформацію

2.3 Постановка завдання

- Збір (генерація) даних: система повинна генерувати дані, що імітують реальну активність користувачів для подальшого аналізу.
- Збір та Агрегація Даних: збір даних з фронтенду та їх агрегація в оперативній базі даних для подальшого аналізу.
- Обробка та Трансформація Даних: виконання ETL-процесів для перетворення зібраних даних в придатний для аналізу формат.
- Аналіз Аномалій та Нотифікація: виявлення аномалій у даних та відправлення нотифікацій через месенджер (наприклад, Telegram).
- Зберігання Даних: вигрузка оброблених даних в сховище даних для довготривалого зберігання та аналізу.
- Аналіз та Прогнозування Даних: завантаження даних в систему для бізнес-аналізу (наприклад, Power BI) для подальшого аналізу та прогнозування.
- Візуалізація Даних: надання інтерфейсу для візуалізації зібраних та оброблених даних.

2.4 Архітектура системи

Система розроблена на основі клієнт-серверної архітектури, що дозволяє забезпечити ефективність, масштабованість, та модульність. Ключовими

компонентами є сервер для зберігання оперативної бази даних (OLTP) і сховища даних (OLAP), а також контейнер Docker з Apache Airflow для ETL-процесів.

Компоненти

- Сервіс Збору Даних: Відповідає за збір і надсилання даних від користувачів до сервера.
- Оперативна БД (OLTP): Сервер, що забезпечує швидкий доступ до зібраних даних.
- Сховище Даних (OLAP): Відповідає за зберігання даних у довготривалому вигляді для аналізу.
- Apache Airflow в Docker: Виконує ETL-процеси для трансформації, насичення, та вигрузки даних в сховище.
- Модуль Доступу: Забезпечує авторизацію та управління доступом для різних ролей (клієнти, аналітики, дата-інженери).

Клієнт-серверна архітектура стала золотим стандартом в розробці розподілених систем через свою гнучкість та ефективність. Вона забезпечує відокремлення бізнес-логіки та інтерфейсу користувача, що спрощує управління та масштабування системи. Ця архітектура також сприяє легшому розподілу ресурсів, оскільки кожен сервер може бути оптимізований для конкретного виду обробки, що в свою чергу підвищує загальну продуктивність системи.

Apache Airflow, в свою чергу, використовує модель директивних ациклічних графів (DAGs) для опису робочих процесів. Це не тільки дозволяє забезпечити високу надійність та консистентність даних, але й спрощує планування завдань та ресурсів. Airflow надає інструменти для моніторингу,

логування, та навіть відновлення після збоїв, що є критично важливим для будь-якої аналітичної системи. [6]

Docker контейнеризація забезпечує додатковий рівень абстракції та автоматизації, дозволяючи легко розгортати та масштабувати компоненти системи. Це дозволяє забезпечити однорідність виконання коду між різними розгортаннями та спрощує процеси CI/CD.

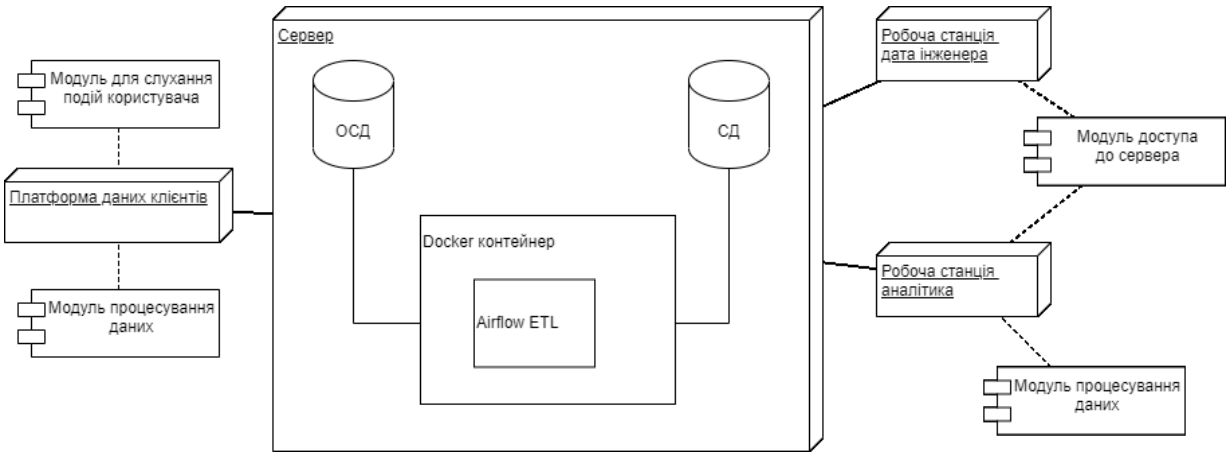


Рис. 3 Топологія системи

2.5 Опис джерела даних

У сховище даних самі дані попадають не структурованими, та мають наступний вигляд

activity_url	monetized	revenue	name_partner	partnership	address	click_ct	view_ct	promo_ct	time_start	time_end	date_event	registration_dt	email	region	active_mapping	birth_dt	credit_score	platform	os_version
example.com/lst_proposal	1	1	Curtis, White and Wilkerson	1	1262 Anthony Islands Suite 323 North Elizabeth, AZ...	3	2	2	12:04:00.00000	12:13:00.00000	2022-10-19	2023-01-24	brownjeremy@example.com	Canada	1	1987-02-02	Fair	Android	12.1
example.com/articles	1	1	Wilkerson LLC	1	9945 Bennett Springs Riverside, CO 22636	0	2	1	17:29:00.00000	17:50:00.00000	2022-10-19	2023-05-18	edwardsaaron@example.net	Alaska	1	1998-09-14	Good	Android	12
example.com/articles	0	2	Cunningham, Howell and Stewart	1	152 Erik Groves South Douglasland, LA 28438	0	1	1	15:07:00.00000	15:10:00.00000	2022-10-19	2022-11-22	hernandezstephen@example.org	Texas	1	1993-03-07	Fair	iOS	17.02
example.com/promotions	1	4	Boyd, Powers and Atkinson	1	PSC 6576, Box 4939 APO AE 31093	0	2	1	23:17:00.00000	23:35:00.00000	2022-10-19	2022-04-15	michael06@example.org	Alaska	1	2005-10-03	Good	iOS	16.7
example.com/articles	1	0	Moses-Orozco	1	654 Glenn Rest West Marychester, IL 49683	0	3	0	19:35:00.00000	19:40:00.00000	2022-10-19	2023-04-17	ehamson@example.com	Texas	1	1981-05-13	Fair	iOS	Invalid Version
example.com/lst_proposal	1	2	Moses-Orozco	1	654 Glenn Rest West Marychester, IL 49683	0	6	0	06:06:00.00000	06:21:00.00000	2022-10-19	2023-04-15	kberg@example.com	Canada	1	1965-11-01	Fair	iOS	15.7
example.com/lst_proposal	1	3	Roberts, Cabrera and Hernandez	1	814 Johnson Islands North Robertfort, GA 50540	2	4	0	13:16:00.00000	13:17:00.00000	2022-10-19	2023-03-05	andrew41@example.com	Canada	1	2003-04-29	Fair	iOS	15.7

Рис. 4 «Сирі» дані

Після трансформації та структуризації структура СД виглядає наступним чином

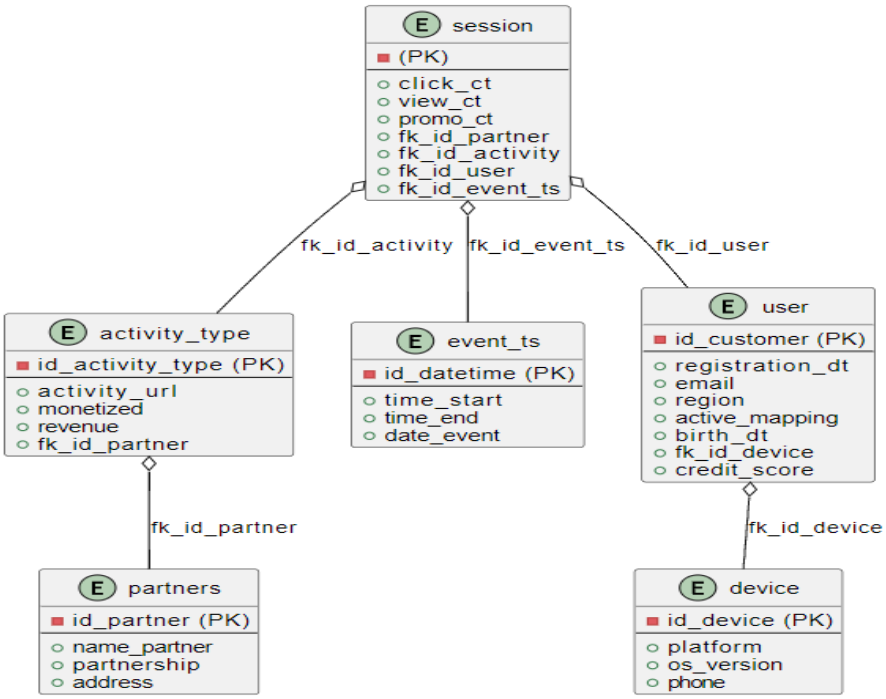


Рис. 5 Сховище даних

- Користувач (customer)
  - дата регістрації
  - емейл
  - регіон
  - активний юзер
  - дата народження
- Рівень кредитної довіри (credit_score)
  - Назва рівня
  - Мінімальна оцінка для рівня
  - Максимальна оцінка для рівня
- Пристрій (device)
  - Платформа
  - Версія платформи
- Партнери (partners)
  - Назва партнеру

    - Співпраця
    - Адреса
- Тип активності (activity_type)
    - Назва активності
    - Місце (посилання на сторінку) активності
    - Активність монетизації
    - Прейскурант активностей
- Сессія (session) – таблиця факт
    - Кількість кліків
    - Кількість переглядів
    - Кількість переходів по акції

3.РОЗРОБКА СИСТЕМИ

3.1 Організаційна структура програмного забезпечення

В системі існує спеціалізований сервіс для збору даних, який взаємодіє з різними джерелами через API. Цей сервіс працює в реальному часі і може отримувати інформацію з веб-сайтів, мобільних додатків, серверів та інших платформ. Після отримання даних, сервіс трансформує їх до потрібного формату, проводячи перевірку агрегацію. Далі, він використовує API для надсилання цих даних до SQL Server, де вони зберігаються в оперативній базі даних для подальшої обробки. Слідуючи кращим практикам обробки потокових даних, сервіс збирає дані та поміщає їх у внутрішню чергу для подальшої обробки. Він може використовувати розподілені системи обробки потокових даних для масштабування та для забезпечення високої пропускної здатності.

Після того, як дані потрапили в чергу, ініціюється їхнє оброблення, яке включає в себе ряд кроків. По-перше, дані валідуються для відповідності заданим

схемам та форматам. По-друге, проводиться очищення даних від шуму та виправлення будь-яких аномалій. По-третє, дані можуть бути збагачені додатковою інформацією або метаданими.

Одразу після обробки, сервіс використовує API для передачі даних до оперативної бази даних на SQL Server. Зазвичай, для цього використовується захищене з'єднання, яке гарантує конфіденційність та цілісність переданих даних.

Важливо відзначити, що весь цей процес є автоматизованим і конфігурується згідно з бізнес-правилами та потребами. Таким чином, сервіс може ефективно адаптуватися до змінних умов та вимог. Згідно з регулятивними актами та дослідженнями в області інформаційної безпеки, недозволене розголошення конфіденційної інформації може призвести до юридичних санкцій. Наприклад, згідно з Загальним регламентом про захист даних (GDPR) Європейського Союзу, порушення конфіденційності даних може призвести до штрафів у розмірі до 4% від глобального річного обороту компанії або 20 мільйонів євро, залежно від того, що більше.[7]

Враховуючи ці санкції та юридичні наслідки, важливо підходити до обробки та зберігання конфіденційної інформації з особливою увагою. Тому в даному проекті було використано генерацію даних для імітації реальних сценаріїв, замість використання реальних користувацьких даних.

Після збору та обробки, сервіс передає дані до оперативної бази даних на SQL Server. Як уже було згадано раніше, дані користувачів є конфіденційними.

Контейнер для управління робочими процесами організовує робочі процеси, пов'язані з перевіркою якості даних, їх трансформацією, насиченням та вивантаженням у сховище даних. Тут знаходять застосування принципи управління робочими процесами, такі як DAGs, які використовуються в системах типу Apache Airflow для моделювання та виконання робочих процесів.

Доступ до платформи надається через модуль доступу, який служить інтерфейсом для клієнтів. Цей модуль надає засоби для створення аналітичних звітів, візуалізації даних, а також реалізує механізми безпечного доступу до ресурсів системи.

Весь дизайн системи базується на клієнт-серверній архітектурі, яка дозволяє забезпечити модульність та віддалений доступ. Ця модель є особливо ефективною для розподілу завдань та ресурсів між серверами та клієнтами, проте в даному випадку не передбачає обслуговування великої кількості користувачів, що робить її ідеально підходящою для даного проекту.

3.2 Вибір інструментарію для створення програмного забезпечення

Підбір оптимального інструментарію є критичним етапом в процесі розробки будь-якої складної інформаційної системи. Вибір конкретних технологій та інструментів впливає на ефективність реалізації проекту, його масштабованість, безпеку, а також на зручність подальшої підтримки. У цьому розділі буде розглянуто інструментарій, що був вибраний для реалізації даного проекту, включаючи серверні та клієнтські технології, бази даних, бібліотеки, а також середовища розробки (IDE). Вибір середовища розробки (IDE)

Для розробки SQL-скриптів використовувався Microsoft SQL Server Management Studio (SSMS). Це інтегроване середовище, яке спеціалізується на управлінні, конфігурації, моніторингу та розробці всього, що пов'язано з Microsoft SQL Server. Вибір цього IDE обумовлений кількома факторами:

- Інтеграція з SQL Server: SSMS розроблено спеціалізовано для роботи з SQL Server, що забезпечує високий рівень інтеграції та спрощує ряд задач.
- Функціональність: Інструмент надає широкий спектр функцій для роботи з базами даних, включаючи засоби для проектування баз

даних, написання SQL-запитів, аналізу продуктивності, дебагінгу та ін.

- Безпека та надійність: SSMS має розширені засоби для управління безпекою, що є критично важливим для будь-якої системи, що працює з конфіденційними даними.
- Підтримка спільноти та документація: Оскільки SSMS є відомим та широко використовуваним інструментом, існує багато ресурсів, що можуть допомогти при розробці, включаючи офіційну документацію, форуми, туторіали та ін.

Таким чином, вибір Microsoft SQL Server Management Studio як IDE для розробки SQL-скриптів був обумовлений його специфічними можливостями, що ідеально підходять для потреб даного проекту.

PyCharm: Для розробки скриптів на Python, в тому числі для роботи з Apache Airflow та іншими бібліотеками, вибрано середовище розробки PyCharm. Вибір цього конкретного IDE обумовлений кількома ключовими перевагами:

- Інтеграція з Python-бібліотеками: PyCharm пропонує вбудовану підтримку багатьох популярних Python-бібліотек, що спрощує процес розробки та тестування.
- Могутні засоби дебагінгу: IDE має розширені інструменти для налагодження коду, що є незамінними при розробці складних ETL-процесів.
- Версійний контроль: Інтеграція з системами версійного контролю, такими як Git, дозволяє ефективно управляти кодом та співпрацювати з іншими розробниками.
- Функціональність та гнучкість: PyCharm надає багатий набір інструментів для рефакторингу коду, автоматичної генерації коду,

інтелектуального код-комплітації та інших функцій, що підвищують продуктивність розробника.

- Підтримка різних типів проектів: Окрім Python, PyCharm підтримує роботу з різними веб-технологіями, форматами даних та базами даних, що дозволяє використовувати одне середовище для різних аспектів проекту.

З урахуванням цих факторів, PyCharm виявився оптимальним вибором для розробки на Python в рамках даного проекту.

Для збору та первинної обробки даних в даному проекті використовується інструмент, який функціонально аналогічний Segment. Цей інструмент дозволяє автоматизувати збір даних з різних джерел та їх подальшу передачу до оперативної бази даних на SQL Server. Основні переваги такого підходу полягають у гнучкості, масштабованості та можливості інтеграції з широким спектром інших систем та платформ.

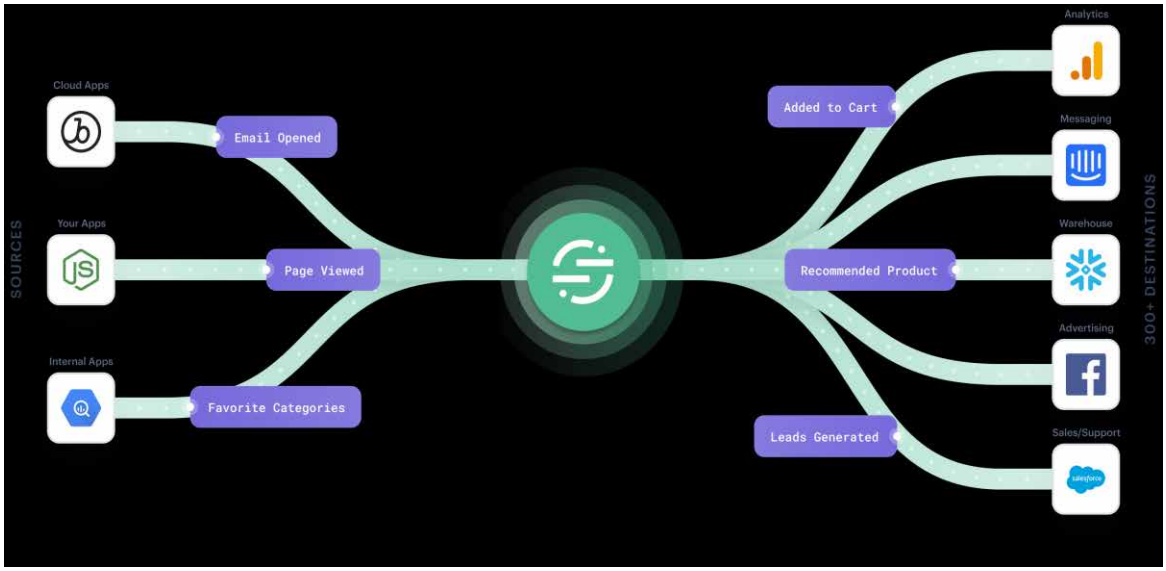


Рис. 6 Потік даних з Segment

За допомогою Segment реалізовано робочий процес збору, обробки і трансферу даних, що базується на подіях. Кожна подія, зареєстрована в системі-джерелі (наприклад, клік користувача, заповнення форми тощо), ініціює API-

виклик для передачі цієї інформації. API є надзвичайно гнучким і дозволяє кастомізацію даних, які потрібно збирати, включаючи користувацькі події та атрибути.

Після отримання події, інструмент проводить її валідацію та можливу трансформацію. Наприклад, можлива фільтрація нецільових подій, збереження лише потрібних атрибутів, чи збагачення даними з інших джерел. Після цього дані поміщаються в чергу для подальшого трансферу.

Система реалізує механізм керування чергою, що дозволяє ефективно управляти великим обсягом даних. Події з черги передаються в оперативну базу даних в пакетному режимі, що забезпечує високу швидкість та надійність передачі.

Трансфер даних може бути налаштований на різні часові інтервали. Наприклад, в режимі реального часу для критичних подій, чи раз на певний часовий інтервал для менш важливих даних. Це дозволяє оптимізувати ресурси та забезпечує гнучкість системи.

Таким чином, використання цього інструменту для збору та передачі даних є ефективним рішенням, що забезпечує швидкість, надійність та масштабованість.

Як було зазначено раніше, для моделювання реалістичних даних використовується генерація на основі Python-бібліотек Faker, random, та numpy.

- Faker дозволяє генерувати великі об'єми реалістичних тестових даних, таких як імена, адреси, номери телефонів тощо.
- Random використовується для генерації випадкових чисел у заданих межах, що може бути корисним для моделювання непередбачуваних подій або змін у даних.

- Numpy надає широкий спектр функцій для наукових розрахунків та обробки масивів даних, що може бути використано для генерації складних статистичних моделей або симуляцій.

Ці бібліотеки дозволяють генерувати даних, які наближені до реальних, з врахуванням різних аспектів та вагових коефіцієнтів, що забезпечує високу якість подальшого аналізу та прогнозування.

В Apache Airflow основний акцент робиться на роботі з розкладами та автоматизації ETL-процесів (Extract, Transform, Load). В цьому контексті, Airflow виступає в ролі планувальника задач, що управляє процесом переносу даних з оперативного сховища в аналітичне, виконує перевірку якості даних, їх трансформацію та насичення.

Для процесів по трансформації, перевіркою якості та перенесення даних був використан інструмент Apache Airflow, у якомуц використовується поняття "DAG" (Directed Acyclic Graph). DAG представляє з себе граф завдань, який визначає, як саме буде виконуватися ETL-процес. Кожен вузол графа відповідає конкретному завданню, а края вказують на залежності між ними.

Для підключення до SQL Server використовується pymssql, що є Python бібліотекою для роботи з Microsoft SQL Server. Ця бібліотека дозволяє взаємодіяти з SQL Server через простий Python API, що значно полегшує процес розробки та підтримки.

Також важливим є те, що Apache Airflow дозволяє інтегрувати в себе додаткові бібліотеки та інструменти для роботи з даними. Наприклад, можливе використання бібліотек Pandas для трансформації даних або TensorFlow для впровадження моделей машинного навчання прямо в ETL-процес.

Apache Airflow був створений для розв'язання конкретних проблем управління розкладами та автоматизації ETL. Він дозволяє реалізувати складні ETL-процеси з високим рівнем гнучкості та контролю. В основі Airflow лежить

концепція коду як конфігурації, що означає, що всі ETL-процеси описуються за допомогою коду, а не через графічний інтерфейс. Це дозволяє забезпечити високий рівень параметризації, повторного використання коду та версіонування.[8] Однією з ключових переваг Apache Airflow є його open-source ліцензія. Це не тільки знижує вартість впровадження та підтримки, але і забезпечує високий рівень гнучкості та налаштуваності. Якщо виникає потреба в специфічних функціональних можливостях, які не підтримуються "з коробки", можна легко додати їх самостійно або скористатися внесками спільноти. Open-source характер також сприяє швидкому розвитку та вдосконаленню продукту, оскільки розробники з усього світу можуть внести свій вклад в проект. Це робить Apache Airflow адаптивним до змінюючихся потреб та технологічних трендів – що дуже підійте даному нетривіальному проєкту.

Таким чином, використання Apache Airflow в системі є обгрунтованим вибором для автоматизації та оптимізації ETL-процесів.

Docker є необхідним компонентом в архітектурі системи, відіграючи ключову роль у контейнерізації сервісів та застосунків. Його використання обґрунтоване декількома важливими перевагами. По-перше, Docker дозволяє ізолювати застосунки та їх залежності в контейнерах, що спрощує розгортання та масштабування. Це особливо корисно для роботи з Apache Airflow, який має ряд залежностей та потребує специфічних конфігурацій.

По-друге, Docker сприяє стабільності системи. Завдяки контейнерізації, можна легко відновити систему в разі збоїв або непередбачуваних проблем. Це забезпечує високу доступність та надійність рішення, яке є критично важливим для бізнес-аналітики та прогнозування.

По-третє, Docker полегшує процес розробки та тестування. Розробники можуть працювати в умовах, які ідентичні продуктивному середовищу, що мінімізує ризики пов'язані з розгортанням нового коду.

Крім того, Docker дозволяє використовувати Docker Compose для опису та запуску множинних контейнерів, що є корисним для комплексних систем, як ваша. В цьому контексті, Docker Compose може бути використаний для оркестрації контейнерів Apache Airflow, SQL Server та інших служб.

У цілому, вибір Docker як інструменту для контейнерізації відповідає потребам сучасних, гнучких та масштабованих архітектур, і є виправданим в рамках даного проекту.

Для нотифікацій щодо якості даних я використав Telegram, але чому, наприклад, не SMTP? Вибір Telegram API для нотифікацій щодо якості даних, на відміну від SMTP (Simple Mail Transfer Protocol), передусім мав обґрунтований ряд причин.

- Швидкість сповіщення: Telegram дозволяє миттєво отримувати повідомлення, що є критично важливим для моніторингу якості даних в реальному часі.
- Взаємодія: Telegram надає можливість не тільки отримувати повідомлення, але і взаємодіяти з ними. Наприклад, можна швидко відповісти на повідомлення, якщо потрібно вжити додаткових заходів.
- Безпека: Telegram використовує сильне шифрування end-to-end, що забезпечує високий рівень безпеки передачі даних.
- Формат повідомлень: Telegram дозволяє відправляти не тільки текстові повідомлення, але і медіа-файли, що може бути корисно для деталізації інформації.
- API та інтеграція: Telegram API є досить гнучким і добре документованим, що спрощує інтеграцію з існуючою системою.
- Відсутність спам-фільтрів: У випадку з електронною поштою, існує ризик, що важливі повідомлення можуть потрапити в папку "Спам",

що ускладнить моніторинг. З 1 лютого 2024 року Google оновлює свою політику доставки Gmail, щоб посилити контроль над маркетингом електронною поштою. Масові відправники, визначені як ті, хто надсилає понад 5000 електронних листів на обліковий запис Gmail за один день, стикаються з новими вимогами. До них належать необхідність автентифікації електронної пошти, кнопка скасування підписки одним натисканням у всіх комерційних електронних листах і дотримання встановленого порогового значення кількості спаму. Недотримання цих вимог призведе до того, що електронні листи не потраплять до одержувачів Gmail. [8]

- Налаштування сповіщень: У Telegram можна налаштувати групи та канали для сповіщення різних груп користувачів згідно їх ролей та обов'язків.
- Низька вартість: Telegram є безкоштовним сервісом, тоді як SMTP може вимагати додаткових витрат на сервери та обслуговування.

Для аналізу в цьому проекті буде використан Power BI. Power BI надає широкий набір інструментів для візуалізації даних, створення інтерактивних звітів та дашбордів. Цей інструмент був обраний за його гнучкість, можливість легкої інтеграції з різними джерелами даних, включаючи SQL Server, та за його розширені можливості в аналізі та прогнозуванні даних.

Power BI дозволяє використовувати розширений DAX (Data Analysis Expressions) для створення більш складних обчислень та метрик, що є важливим для глибокого бізнес-аналізу. Це додає додаткову цінність до системи, оскільки забезпечує можливість виконання високоякісного аналізу без необхідності інтеграції додаткових інструментів аналітики.

Таким чином, включення Power BI в стек технологій проекту дозволяє не тільки зробити аналіз даних більш ефективним, але і забезпечує можливість

швидкого отримання інформативних висновків для прийняття обґрунтованих бізнес-рішень.

В рамках розробки комплексної системи для збору, обробки, аналізу та візуалізації даних було використано ряд інструментів та технологій, кожен з яких відіграє свою ключову роль.

Середа розробки Microsoft SQL Server Management Studio і PyCharm були вибрані з огляду на їх функціональні можливості, підтримку великої кількості мов програмування та зручність використання.

Інструменти збору та первинної обробки даних були налаштовані таким чином, щоб максимально забезпечити швидкість та надійність збору інформації, з використанням бібліотек Python для генерації тестових даних.

Apache Airflow, як відкритий проект, надає велику гнучкість у налаштуванні процесів ETL.

Docker використовується для контейнерізації, що забезпечує ізоляцію та уніфікацію середовища.

Telegram API слугує для швидких та безпечних нотифікацій, що є важливим у контексті моніторингу якості даних.

З огляду на конфіденційність даних, усі процеси розроблені з урахуванням найвищих стандартів безпеки. На основі джерел, незаконне розголошення конфіденційної інформації може призвести до санкцій, що підкреслює важливість цього аспекту.

В заключенні важливо відзначити роль Power BI у цьому проекті. Цей інструмент було обрано для бізнес-аналізу та прогнозування з метою отримання глибоких в insights зібраних даних. За допомогою Power BI можливо створювати інтерактивні дашборди та звіти, які значно полегшують процес прийняття рішень. Інтеграція з SQL Server та використання мови DAX для складних обчислень роблять Power BI незамінним інструментом в арсеналі даного проекту.

Його гнучкість та масштабованість забезпечують можливість адаптації під конкретні бізнес-вимоги, що робить його вибір обґрунтованим.

В цілому, вибір кожного інструменту був зроблений на основі його специфічних переваг, можливостей інтеграції та з урахуванням потреб проекту. Такий підхід не тільки оптимізує робочий процес, але і забезпечує високий рівень надійності та безпеки системи.

3.3 Алгоритмізація та програмування програмних модулів

Користувач взаємодіє з веб-додатком, переглядаючи різні сторінки та клікаючи на різні пропозиції. Кожна така взаємодія представляє собою івент, який потрібно зафіксувати для подальшого аналізу. Івенти можуть бути різними: перегляд сторінки, клік на пропозицію, перехід на іншу сторінку і так далі.

3.3.1 Реалізація зчитування івентів

На фронтенді можна використовувати специфічні для Segment JavaScript API. Це дозволить відправляти івенти на бекенд для подальшої обробки. Приклад коду на фронтенді:

```
analytics.track('Proposal Click', {
  url: 'example.com',
  Partner: 'ABC',
  session_timestart: '12:15:00',
  session_timeend: '12:30:00',
  date: '2022-10-30',
  region: 'Texas',
  platform: 'Android',
  os_version: '12.1'
});
```

Рис. 7 Приклад використання Segment на фронтенді

Та сама релізація у цьому проєкті є генерацією даних для наповнення системи за допомогою Python, а саме бібліотеки Faker, random та numpy. Faker дозволяє створювати велику кількість реалістичних тестових даних, в той час як random та numpy можна використовувати для генерації числових даних та

симуляції випадкових процесів. Для кожної сутності у СД створено окрему генерацію даних. Ось їх список функцій та реалізація для деяких з них:

- Генерація даних для даних партнерів (дані були згенеровані для 50 партнерів)
- Генерація даних для даних активностей
    1. Кожна активність має партнера, з різною вартістю за кожну з активностей
- Генерація даних пристроїв користувачів
    1. У цій роботі пристрої користувачів обмежаться лише смартфонами на базі Android версій розпочинаючи з 12.0 та на базі iOS версій розпочинаючи з 15.7 (доля iOS складає 60 відсодків, а Android – решта. Саме такі цифри наразі є дійсними у Сполучених Штатах Америки для мобільних девайсів. [10])
- Генерація даних користувачів
    1. Кожен користувач має 3 рівня кредитної довіри: Нормальний, Поганий, Гарний.
    2. Користувачі обмежені територіально, ось перелік штатів: Аляска, Каліфорнія, Нью Йорк, Техас, Онтаріо, Кьюбек (а так не визначений штат позначається просто як США).
    3. Дата народження та емейл є штучно згенерованими та не є справжніми задля запобігання порушення конфіденційності.
- Генерація даних сесії
    1. У кожній сесії користувач робить якісь дії (кліки, перегляди або переходи), для усіх дій я створив ваги

```
promo_weights = [0.6, 0.3, 0.1]
click_weights = [0.4, 0.3, 0.2, 0.1]
view_weights = [0.1, 0.2, 0.3, 0.15, 0.1, 0.05, 0.05, 0.05]
```

Рис. 8 Ваги для генерації даних дій користувача

На рис. 8 зображено ваги до кожного типу дій, де індекс кожного з елемента є кількістю дій користувача (наприклад, на даній платформі користувач в середньому передивляється 2 пропозиції і саме через це число у масиві view_weights індекс 3 елементу є найбільший)

- Генерація часу сесій
    1. Відокремлено в окрему логіку оскільки є досить складним процесом через особливість генерації. Оскільки дані у системи повинні бути якнайближче до реальних, потрібно звертати особливу увагу на розподілення часу та днів тижнів, оскільки частіше за всього трафік є не сталою величиною.
        1. Для кожного дня є ваги

```
day_of_week_weights = np.array([0.17, 0.175, 0.2, 0.125, 0.13, 0.1, 0.11])
```

Рис. 9 Ваги для днів тижня

        2. Окрім генерації днів по вагах на кожен є ще час для кожного дня

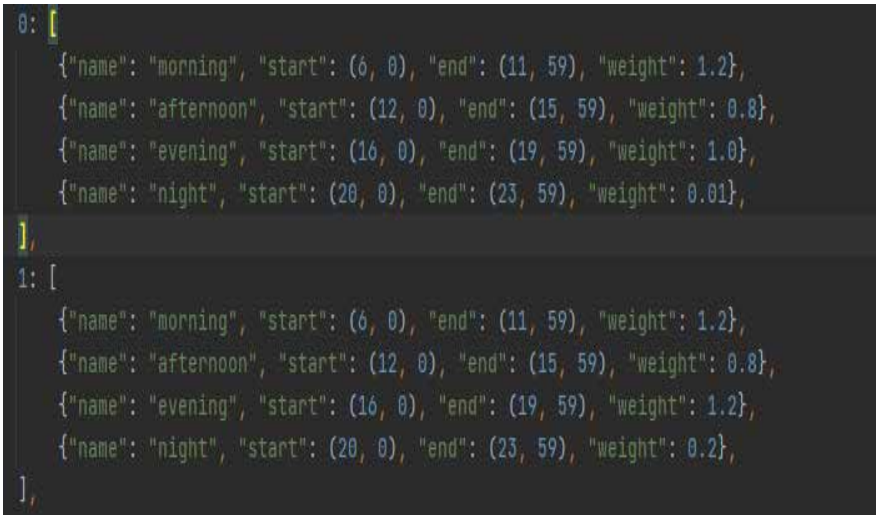


Рис. 10 Ваги для часу

Кожен елемент зображений на рис. 9 є днем тижня з розбитими на 4 пори дня: ранок, день, вечір та ніч, у кожної пори є свій інтервал та вага.

Код для генерації часу сесії

```python
def choose_time_interval(time_intervals, day_of_week):
    intervals = time_intervals[day_of_week]
    return random.choices(intervals, [i["weight"] for i in intervals])[0]


def generate_random_time(interval):
    start_hour, start_min = interval["start"]
    end_hour, end_min = interval["end"]
    random_hour = random.randint(start_hour, end_hour)
    random_minute = random.randint(start_min, end_min)
    return f"{random_hour:02}:{random_minute:02}:00"


def generate_event_ts_data(lookback_days, num_rows):
    data = []
    today = datetime.now().date()
    start_date = today - timedelta(days=lookback_days - 1)
    total_weight = sum(day_of_week_weights)
    normalized_weights = [w / total_weight for w in day_of_week_weights]
    weekday_distribution = [normalized_weights[i % 7] for i in range(lookback_days)]
    total_weekday_weight = sum(weekday_distribution)
    daily_rows = [int(num_rows * w / total_weekday_weight) for w in weekday_distribution]
    remaining_rows = num_rows - sum(daily_rows)
    for i in range(remaining_rows):
        daily_rows[i % len(daily_rows)] += 1
    for offset, daily_quota in enumerate(daily_rows):
        current_date = start_date + timedelta(days=offset)
        weekday = current_date.weekday()
        print(f"Number of rows for {current_date}: {daily_quota}")  # Print daily rows generated
        for _ in range(daily_quota):
            interval = choose_time_interval(time_intervals, weekday)
            time_start = generate_random_time(interval)
            minutes_interval = random.randint(0, 30)
            time_start_datetime = datetime.strptime(f"{current_date} {time_start}", "%Y-%m-%d %H:%M:%S")
            time_end_datetime = time_start_datetime + timedelta(minutes=minutes_interval)
            time_end = time_end_datetime.strftime("%H:%M:%S")
            current_date_formatted = current_date.strftime("%Y-%m-%d")
            data.append((time_start, time_end, current_date_formatted))
    print(f"Generated {len(data)} rows")
    return data
```

Рис. 11 Код функцій які генерують дані для сесії

Варто зазанчити, що окрім цих функцій для генерації часу сесії використовують ще ваги для днів та проміжків часу зазначених на рис. 9 та рис. 10 відповідно.

Опис коду на зображеного на рис.11

Цей код створює синтетичні дані для генерації подій в часових проміжках. Ось функції:

- choose_time_interval(time_intervals, day_of_week): Вибирає часовий проміжок для конкретного дня тижня, враховуючи ваги.
- generate_random_time(interval): Генерує випадковий час в межах вибраного часового проміжку.
- generate_event_ts_data(lookback_days, num_rows): Генерує велику кількість подій, враховуючи ваги для кожного дня тижня та часових проміжків.

Робота коду

1. Нормалізує ваги днів тижня, щоб їх сума була рівна 1.
2. Розподіляє кількість рядків (num_rows), які потрібно згенерувати, між днями тижня відповідно до нормалізованих ваг.
3. Для кожного дня генерує випадковий час початку та завершення події, а також дату.
4. В результаті роботи функції generate_event_ts_data отримуємо список кортежів, де кожний кортеж містить час початку та завершення події, а також дату цієї події.

3.3.2 Потік даних

Для організації потоку даних було використано Apache Airflow, яка є платформою для програмування, планування та моніторингу робочих процесів. Його основна перевага полягає у можливості декларативно описувати робочі процеси за допомогою Python, що дозволяє легко інтегрувати його з іншими інструментаи і сервісами. Airflow підтримує розподілений режим роботи, де ви можете планувати та моніторити завдання на кластері з багатьох вузлів. За рахунок своєї гнучкості та розширюваності, Airflow став одним із ключових інструментів для ETL-процесів в сучасних дата-інжинірингових підходах. [11]

Apache Airflow використовується для планування та виконання ETL-завдань. Завдання зчитують дані з оперативного сховища даних (ОСД),

перевіряють їх на якість(з надсиланням повідомлень про наявність проблем), трансформують та вивантажують у кінцеве сховище даних.

До процесу ETL, реалізованого за допомогою Apache Airflow, включено етап перевірки якості даних. При цьому, якщо кількість помилок у даних досягає певного порогового рівня, система автоматично надсилає повідомлення в Telegram. Це реалізовано за допомогою Telegram API, що інтегровано в Airflow DAG.

Такий підхід забезпечує оперативне інформування команди про проблеми з якістю даних, дозволяючи швидко вжити коректив них заходів. Це особливо важливо для забезпечення надійності аналітичних моделей та звітів, які генеруються на основі цих даних.

В нашому випадку, Apache Airflow розгортається всередині Docker контейнера. Це забезпечує декілька ключових переваг:

1. Ізоляція середовища: Docker контейнер ізолює Airflow від інших процесів та залежностей на хост-машині. Це забезпечує стабільність роботи.
2. Переносимість: Завдяки контейнерізації, всі залежності та конфігурації пакуються разом з самим Airflow, що дозволяє легко переносити ETL процеси між різними машинами або середовищами.
3. Масштабованість: Docker дозволяє легко масштабувати Airflow, додавання нових вузлів до кластеру або змінюючи ресурси виділені для контейнера.
4. Контроль версій: Використання Docker дозволяє тримати різні версії Airflow, що може бути корисним для тестування нових фіч або змін та використання на різних типах середовища (тестове, стейджове або виробниче)

5. Безпека: Ізоляція контейнера також додатково підвищує рівень безпеки, оскільки потенційні уразливості в одному сервісі не впливають на інші.

Використання Docker для розгортання Apache Airflow є частиною загальної стратегії контейнерізації сервісів в даному проєкті, що забезпечує гнучкість та ефективність управління ресурсами.

В Apache Airflow для реалізації різних аспектів управління потоками даних використовуються наступні типи воркерів:

- Webserver: Цей компонент відповідає за веб-інтерфейс Airflow. Через нього користувачі можуть моніторити стан задач, запускати та зупиняти потоки даних, конфігурувати параметри задач і так далі.
- Scheduler: Scheduler відповідає за планування виконання задач на основі визначених у DAG трігерів і залежностей між задачами. Цей компонент перевіряє DAG на предмет готовності задач до виконання і передає їх до Worker'ів.
- Worker: Worker'и фактично виконують задачі, які їм доручає Scheduler. Вони можуть бути розміщені на одній або декількох машинах для паралельного виконання задач.
- Trigger: Це не стандартний воркер Airflow, але в контексті певних конфігурацій може бути реалізована логіка для ініціювання потоків даних на основі певних подій або умов.
- Init: Це воркер, який ініціалізує базу даних та інші необхідні ресурси перед запуском Airflow. Зазвичай використовується в контейнерізованих розгортаннях.

- Postgres: Це не прямий воркер Airflow, але важливий компонент, який служить базою даних для зберігання метаданих, стану задач, конфігурацій і так далі.
- Redis: Це також не воркер в строгому розумінні, але важливий сервіс для кешування та черги повідомлень. Він може бути використаний для зберігання проміжних даних і координації між Worker'ами для паралельного виконання задач.

Рис. 12 Запищений контерней Airflow у Docker Desktop

Ці компоненти були розгорнуті (див рис. 12) в одному Docker-контейнерах для забезпечення масштабування, відмовостійкості та ізоляції процесів. Наприклад, запуск Airflow в Docker контейнерах дозволяє легко управляти ресурсами, здійснювати розгортання та забезпечувати ізоляцію середовища.

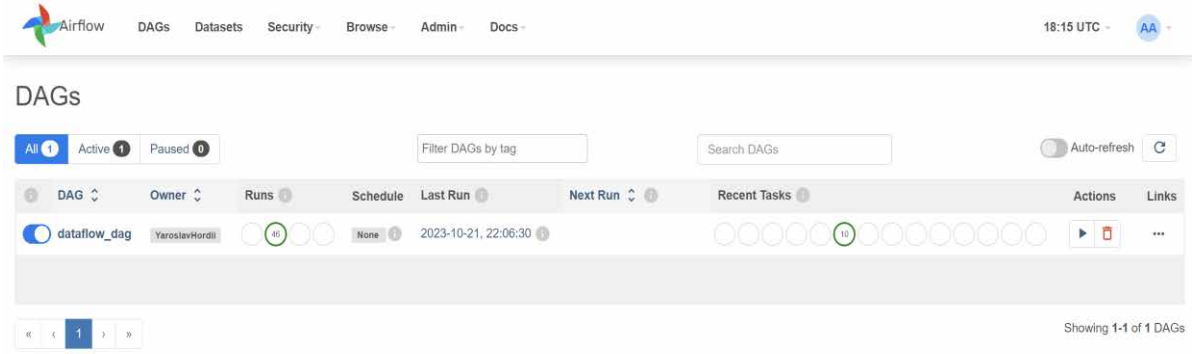


Рис. 13 Інтерфейс головної сторінки Apache Airflow

На рис. 12 ми бачимо задіяний порт 9099 для воркера webserver який є репрезентацією інтерфейсного функціоналу Airflow, а через те що сервером в цій роботі виступає локальний комп’ютер то доступ до цього веб серверу здійснюється за допомогою посилання localhost:9099, що є на головній сторінці цього веб серверу зображено на рис. 13: тут ми бачимо список DAGs (в нашому випадку його назва dataflow_dag) – у кожного є перемикач, назва, власник, кількість запусків та їх успішність, розклад, час та дата останнього запуску, запланований запуск, останні таски, кнопка запуску та видалення. Також присутні настройки цього DAG.

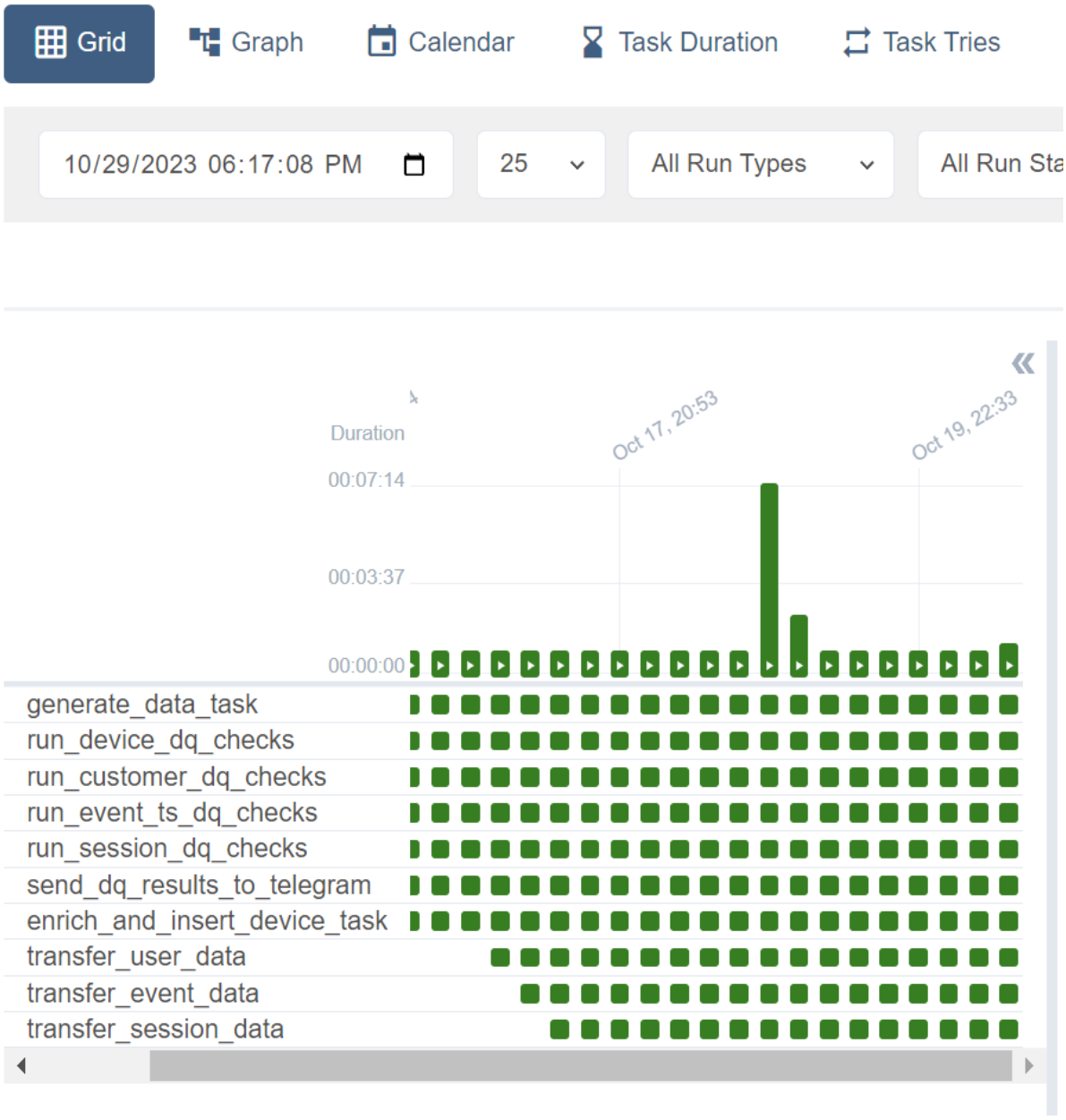


Рис. 14 Таблиця успішності тасок у DAG в розділі Grid

При переході на один з них ми побачимо таблицю з успішністю останніх запусків по кожній тасці.

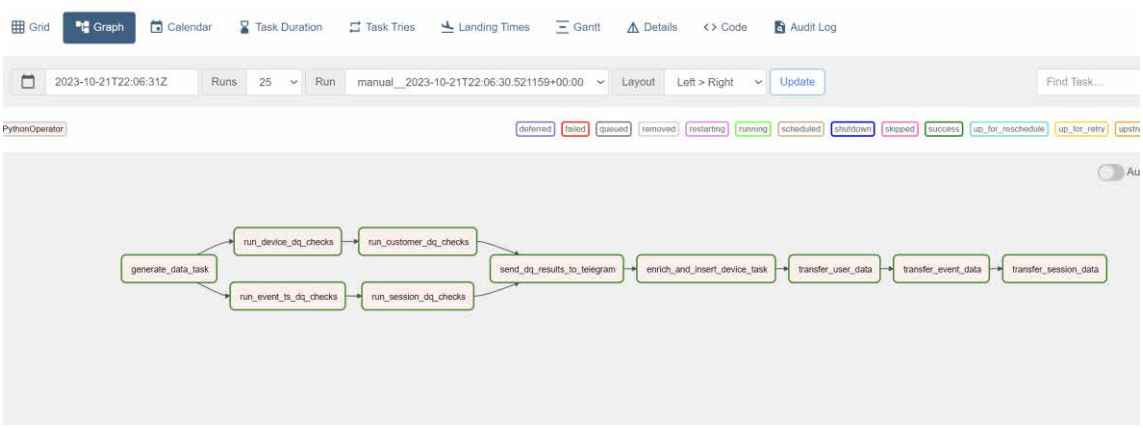


Рис. 15 Граф з черговістю та успішністю за останній запуск

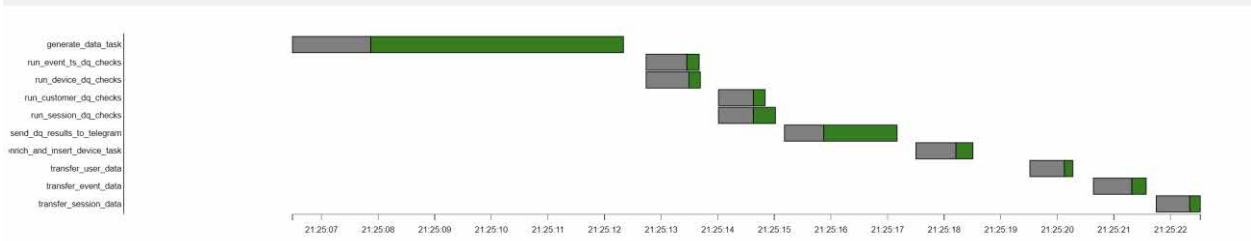


Рис. 16 Час та послідовність завдань

У розділі Graph ми вже побачимо черговість виконання завдань та результати їх роботи за останній запуск (див рис. 15) а вже на рис. 16 їх черговість поєднана з часом виконань кожної з них.

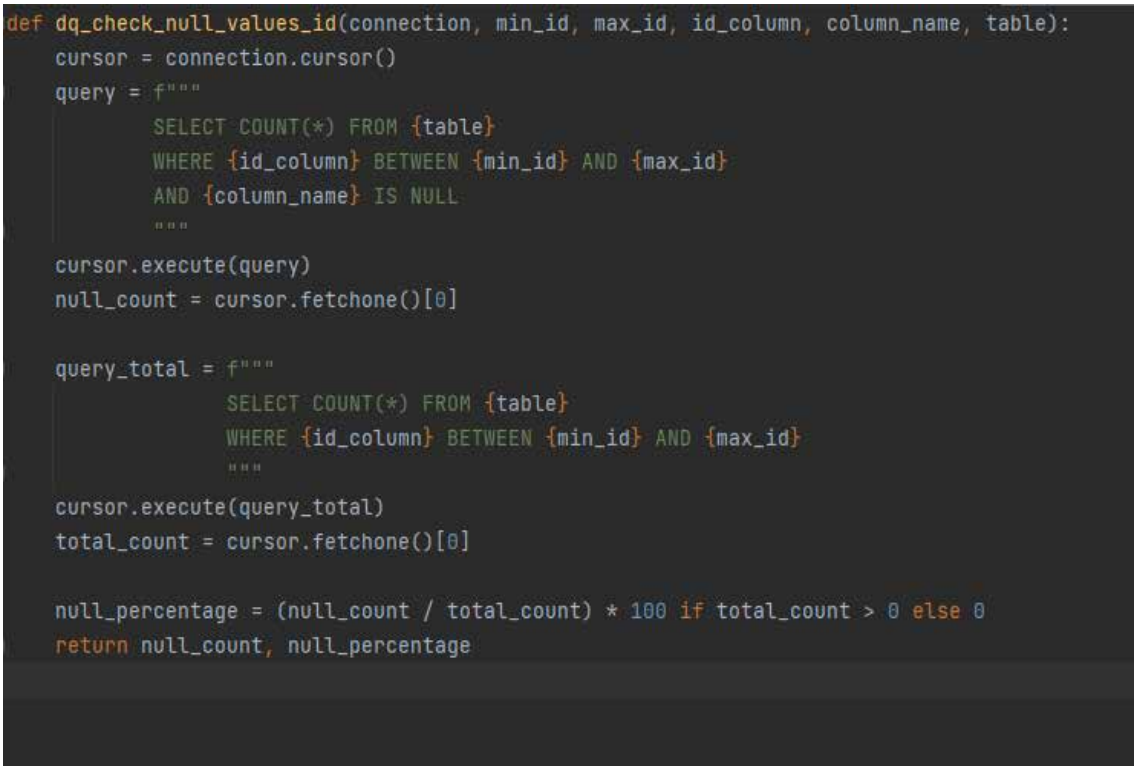
Зупинимось на тасках (або завданнях) детальніше та розберемо їх код та виклик у самому DAG

1. generate-data-task – завдання, код та реалізація якого була розглянута у розділі 3.3.1
2. run_device_dq_check (перевірка якості даних девайсу), run_customer_dq_checks (перевірка якості даних користувачів), run_event_ts_dq_checks (перевірка якості часових даних), run_session_dq_checks (перевірка якості даних сесії) – ті самі перевірки якості даних які я реалізував як завдання в середині виконання ETL процесу

1. run_device_dq_check – дві перевірки: нестандартна платформа, не числове значення версії
2. run_customer_dq_checks - дві перевірки: формат емейлу, майбутня дата реєстрації
3. run_event_ts_dq_checks – три перевірки: перевірка на відсутність даних, тривалість сесії, час кінця сесії не менший за початок
4. run_session_dq_checks – дві перевірки: аномальна кількість дій, кількість кліків та переходів при 0 переглядів

3. send_dq_results_to_telegram – при наявності відхилень якості даних надсилає повідомлення у телеграм (при їх відсутності просто переходить до наступного завдання)
4. enrich_and_insert_device – додає колонку девайсу з назвою моделі телефону (реалізація за допомогою AWS Lambda однак поза рамками цієї роботи) та переносить дані у СД
5. transfer_user_data – переніс даних користувачів фінансової установи у СД
6. transfer_event_data – переніс часових даних у СД
7. transfer_session_data – переніс даних сесій у СД

Приклад реалізації деяких з завдань



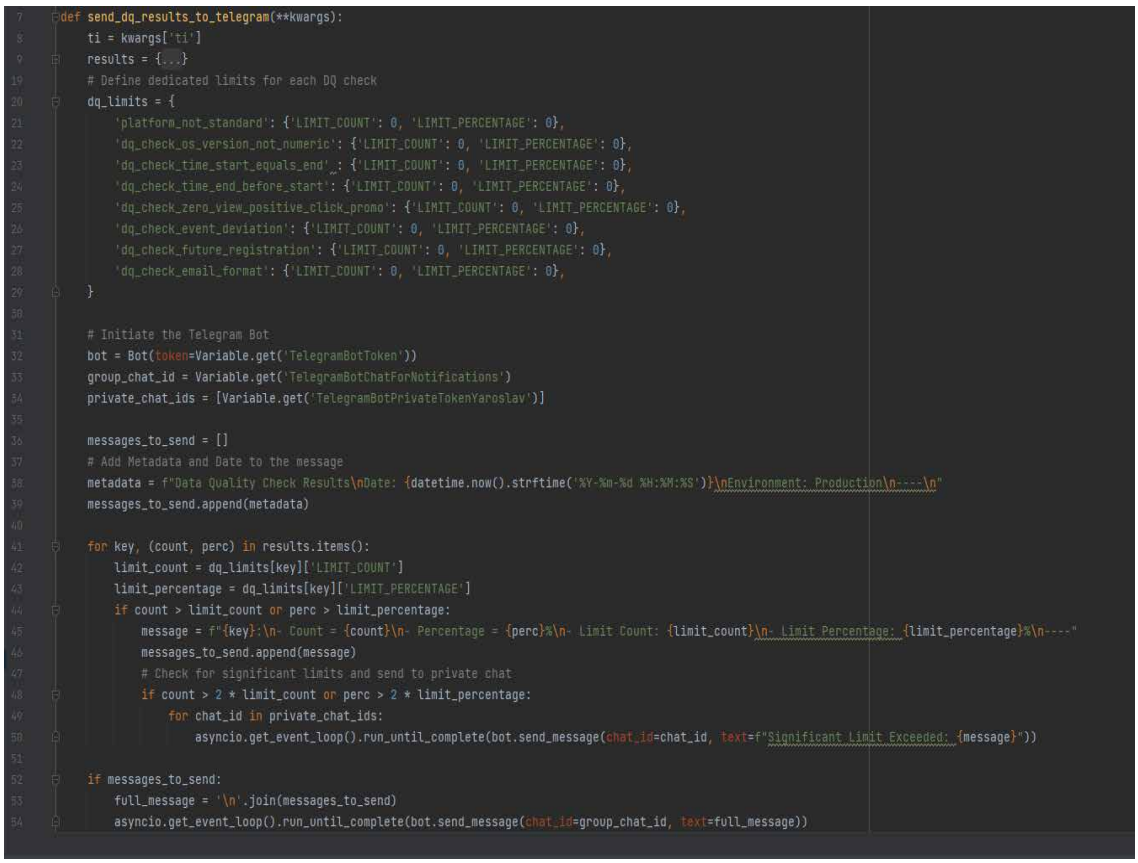
```
def dq_check_null_values_id(connection, min_id, max_id, id_column, column_name, table):
    cursor = connection.cursor()
    query = f"""
            SELECT COUNT(*) FROM {table}
            WHERE {id_column} BETWEEN {min_id} AND {max_id}
            AND {column_name} IS NULL
            """
    cursor.execute(query)
    null_count = cursor.fetchone()[0]

    query_total = f"""
                SELECT COUNT(*) FROM {table}
                WHERE {id_column} BETWEEN {min_id} AND {max_id}
                """
    cursor.execute(query_total)
    total_count = cursor.fetchone()[0]

    null_percentage = (null_count / total_count) * 100 if total_count > 0 else 0
    return null_count, null_percentage
```

Рис. 17 Реалізація загальної функції для перевірки нулей у таблиці

На рис. 17 зображено завдання на перевірку кількості нулей у таблиці. У сигнатурі методи маємо наступне: connection (підключення до ОСД), min_id та max_id (інтервал даних для перевірки), id_column (колонка ідентифікатор), column_name (колонка на перевірка), table (таблиця на перевірку). При виклику цієї функції та передачі усіх параметрів поверне кількість рядків з нулями та відсоткову частину нулів від усіх строк. Ця перевірка використовується у завдання run_event_ts_dq_checks.



```python
def send_dq_results_to_telegram(**kwargs):
    ti = kwargs['ti']
    results = {...}
    # Define dedicated limits for each DQ check
    dq_limits = {
        'platform_not_standard': {'LIMIT_COUNT': 0, 'LIMIT_PERCENTAGE': 0},
        'dq_check_os_version_not_numeric': {'LIMIT_COUNT': 0, 'LIMIT_PERCENTAGE': 0},
        'dq_check_time_start_equals_end': {'LIMIT_COUNT': 0, 'LIMIT_PERCENTAGE': 0},
        'dq_check_time_end_before_start': {'LIMIT_COUNT': 0, 'LIMIT_PERCENTAGE': 0},
        'dq_check_zero_view_positive_click_promo': {'LIMIT_COUNT': 0, 'LIMIT_PERCENTAGE': 0},
        'dq_check_event_deviation': {'LIMIT_COUNT': 0, 'LIMIT_PERCENTAGE': 0},
        'dq_check_future_registration': {'LIMIT_COUNT': 0, 'LIMIT_PERCENTAGE': 0},
        'dq_check_email_format': {'LIMIT_COUNT': 0, 'LIMIT_PERCENTAGE': 0},
    }

    # Initiate the Telegram Bot
    bot = Bot(token=Variable.get('TelegramBotToken'))
    group_chat_id = Variable.get('TelegramBotChatForNotifications')
    private_chat_ids = [Variable.get('TelegramBotPrivateTokenYaroslav')]

    messages_to_send = []
    # Add Metadata and Date to the message
    metadata = f"Data Quality Check Results\nDate: {datetime.now().strftime('%Y-%m-%d %H:%M:%S')}\nEnvironment: Production\n----\n"
    messages_to_send.append(metadata)

    for key, (count, perc) in results.items():
        limit_count = dq_limits[key]['LIMIT_COUNT']
        limit_percentage = dq_limits[key]['LIMIT_PERCENTAGE']
        if count > limit_count or perc > limit_percentage:
            message = f"{key}:\n- Count = {count}\n- Percentage = {perc}%\n- Limit Count: {limit_count}\n- Limit Percentage: {limit_percentage}%\n----"
            messages_to_send.append(message)
            # Check for significant limits and send to private chat
            if count > 2 * limit_count or perc > 2 * limit_percentage:
                for chat_id in private_chat_ids:
                    asyncio.get_event_loop().run_until_complete(bot.send_message(chat_id=chat_id, text=f"Significant Limit Exceeded: {message}"))

    if messages_to_send:
        full_message = '\n'.join(messages_to_send)
        asyncio.get_event_loop().run_until_complete(bot.send_message(chat_id=group_chat_id, text=full_message))
```

Рис. 18 Реалізація завдання відправки повідомлень Телеграм

На рис. 18 зображено код завдання send_dq_results_to_telegram, який в якості змінних отримує результати аналізу якості даних та в подальшому підставляючи ліміти під кожен з них вирішує відправляти їх чи ні. Якщо аналіз якості даних переходить межі то спочатку нотифікації відправляються у групу, де знаходяться кілька дата інженерів, якщо ж помилки в якості даних більш ніж у два рази перевищують ліміти то повідомлення надходять у группу а також у приватні повідомлення зазначеним особам (у коді на рис. 17 це ідентифікаційні номери чатів груп та приватних повідомлень розміщених на рядках 33 та 34 відповідно). Через безпекові обмеження та зручності встановлення через веб інтерфейс, Airflow використовує спеціалізований розділ для зберігання ідентифікаційних номерів та інших конфіденційних даних, званий Variables. Variables — це вбудоване сховище даних всередині самого Airflow, і доступ до цих даних програма має лише під час її виконання. Це особливо корисно для

під’єднання до SQL Server, де інформація для входу також зберігається в цьому сховищі.

В Apache Airflow, механізм змінних (Variables) не лише служить для зберігання конфіденційної інформації, але і надає можливість її шифрування для додаткового рівня безпеки. За замовчуванням, Airflow використовує шифрування на рівні бази даних. Це здійснюється за допомогою Fernet-ключа, який зберігається в конфігураційному файлі airflow.cfg під секцією [fernet_key]. Після налаштування Fernet-ключа, всі змінні, які додаються або оновлюються, будуть автоматично шифровані перед зберіганням в базу даних. Цей механізм шифрування є особливо важливим для забезпечення конфіденційності і інтеграції даних, та дозволяє дотримуватися стандартів безпеки та регулятивних вимог.

Рис. 19 Група та приватні повідомлення від бота Телеграма

На рис. 19 результати знаходження помилок у даних під час, розгянемо їх детальніше вже у наступному розділі.

3.3.3 Аналіз та прогнозування

В контексті архітектури обробки даних, Power BI використовується для бізнес-аналізу та прогнозування. Після завершення всіх процесів збору, перевірки та трансформації даних, останнім етапом є візуалізація цих даних та виведення інсайтів для прийняття бізнес-рішень.

Power BI дозволяє підключити безліч джерел даних, включаючи SQL Server. Зазвичай, підключення відбувається через специфічні конектори, які можна налаштувати для автоматичного оновлення даних. За допомогою різних типів чартів, графіків та дашбордів, Power BI дозволяє складно аналізувати та

інтерпретувати великі об'єми даних. Окрім цього, Power BI має можливості для машинного навчання та прогнозування, які можна використовувати для аналітики в реальному часі.

Використання Power BI в даному проекті є логічним завершенням потоку обробки даних. Співвідношення івентів користувача, аналіз поведінки, вивчення паттернів та прогнозування майбутньої поведінки — все це можливо реалізувати за допомогою даного інструменту. Його гнучкість та здатність інтегруватися з різними джерелами даних, в тому числі SQL Server, робить його незамінним елементом в архітектурі обробки даних.

4. РЕЗУЛЬТАТИ ДОСЛІДЖЕННЯ

4.1 Результати розробки підсистеми обробки та аналізу даних фінансової онлайн установи

Для досягнення результатів дослідження було побудовану підсистему обробки та аналізу даних фінансової онлайн установи, як і зазначалося в минулих розділах в ролі ОСД та СД виступає SQL Server та сама генерація даних є частиною потому даних, де можна встановити вікно днів назад, для яких буде згенеровано дані, а також потрібно вказати кількість потрібних строк. Кількість строк/Кількість днів = приблизна кількість строк на все вікно днів (як вже було сказано, кожен день має певну вагу, тому можливо що строк буде чи менше чи більше, але їх сумма буде дорівнювати зазначеному значенню.

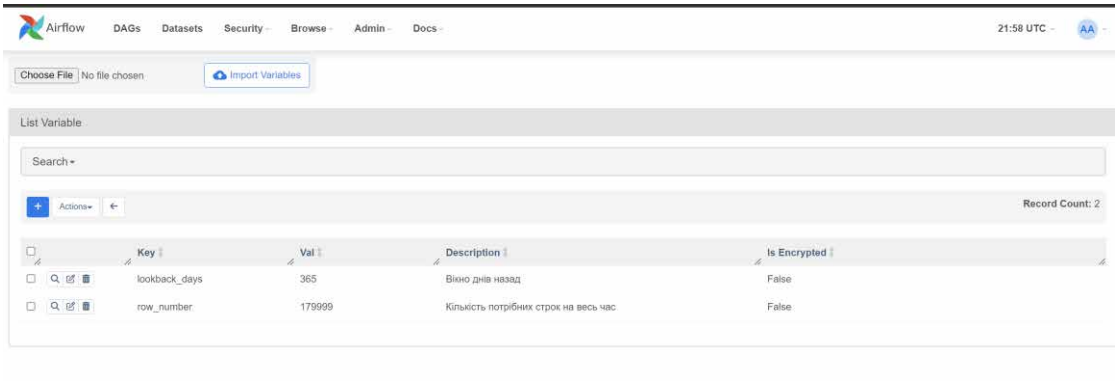


Рис. 20 Сховище змінних Airflow

У сховищі змінних на рис. 20 зазначено такі значення, щоб дані були згенеровані та перевірені на останні 365 днів та згенерували 180000, що в результаті дасть нам приблизно 500 строк генерації на кожен день.

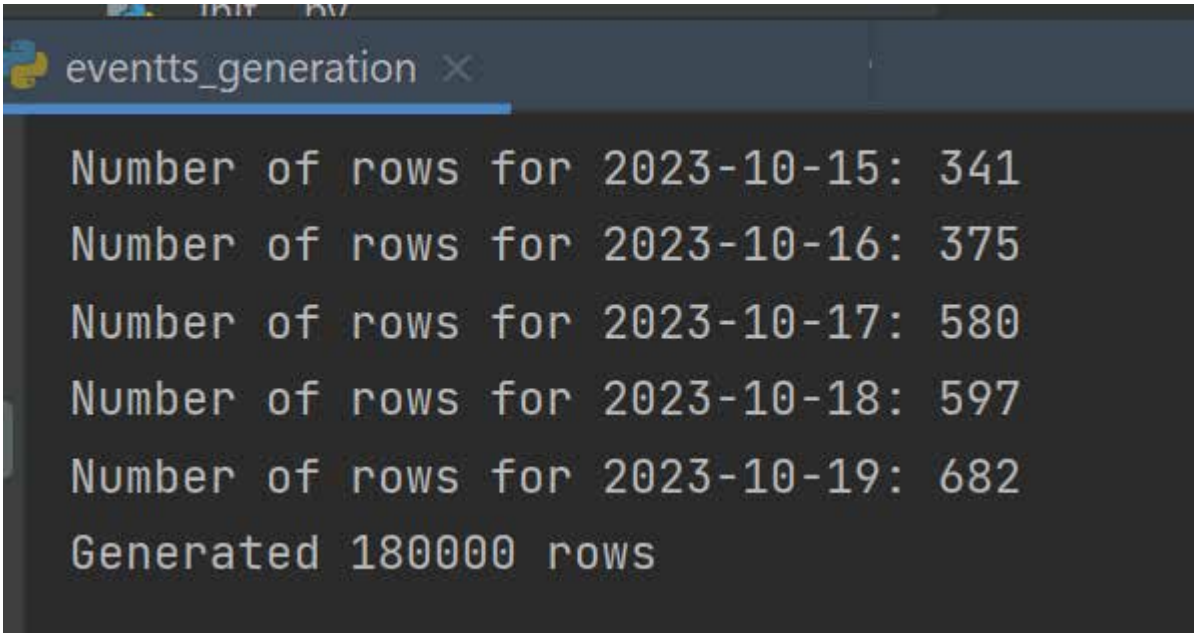


Рис. 21 Генерація даних

```
INSERT INTO [dbo].[event_ts] (time_start, time_end, date_event) VALUES ('21:38:00', '21:51:00', '2023-10-01');
INSERT INTO [dbo].[event_ts] (time_start, time_end, date_event) VALUES ('20:43:00', '20:43:00', '2023-10-01');
INSERT INTO [dbo].[event_ts] (time_start, time_end, date_event) VALUES ('19:58:00', '20:19:00', '2023-10-01');
INSERT INTO [dbo].[event_ts] (time_start, time_end, date_event) VALUES ('15:54:00', '16:19:00', '2023-10-01');
```

Рис. 22 Логи вставки даних у СД

Вже на рис. 21 бачимо результат генерації даних. Наприклад для 2023-10-15 (15 жовтня 2023 рік) маємо лише 341 рядків, коли для 2023-10-19 вже згенеровано 682 рядка і резюмуючи бачимо що всі строки були згенеровані.

```
generate_data_task >> run_device_dq_checks >> run_customer_dq_checks >> send_dq_results_to_telegram
generate_data_task >> run_event_ts_dq_checks >> run_session_dq_checks >> send_dq_results_to_telegram


send_dq_results_to_telegram >> enrich_and_insert_device >> transfer_user_data >> transfer_event_data >> transfer_session_data
```

Рис. 23 Порядок виконання завдань у DAG

Конфігурацію порядку виконання завдань у DAG видно на рис. 23

Для ефективнішої роботи бізнес аналітиків та дата інженерів створимо уявлення для агрегації та подальшого використання даних у СД. Ось деякі з них:

- [MetricsSummary] – збір основних метрік по всіх таблицях, але з деякими відкинутими колонками. Уявлення об'єднує дані з різних таблиць (session, event_ts, [user], device, activity_type, partners) для отримання загальної інформації про користувачів, їхні сесії, дії та дохід. Запит використовує ряд функцій та обчислень для отримання додаткових даних, таких як тривалість сесії та вік користувача.

```sql
CREATE VIEW [dbo].[MetricsSummary] AS
SELECT
    s.view_ct,
    s.click_ct,
    s.promo_ct,
    e.time_start,
    e.time_end,
    DATEDIFF(MINUTE, e.time_start, e.time_end) AS session_time, -- Calculating session time in minutes
    e.date_event,
    u.region,
    DATEDIFF(YEAR, u.birth_dt, GETDATE()) -
        CASE
            WHEN MONTH(u.birth_dt) > MONTH(GETDATE()) OR
                (MONTH(u.birth_dt) = MONTH(GETDATE()) AND DAY(u.birth_dt) > DAY(GETDATE()))
            THEN 1
            ELSE 0
        END AS years_old,
    u.registration_dt,
    d.platform,
    d.os_version,
    d.phone,
    COALESCE(
      SUM(CASE WHEN at.activity_name = 'click_proposal' THEN s.click_ct * at.revenue ELSE 0 END), 0
    ) +
    COALESCE(
      SUM(CASE WHEN at.activity_name = 'view_proposal' THEN s.view_ct * at.revenue ELSE 0 END), 0
    ) +
    COALESCE(
      SUM(CASE WHEN at.activity_name = 'click_promo' THEN s.promo_ct * at.revenue ELSE 0 END), 0
    ) AS total_revenue,
    p.name_partner,
  s.team
FROM session s
JOIN event_ts e ON s.fk_id_event_ts = e.id_datetime
JOIN [user] u ON s.fk_id_user = u.id_customer
JOIN device d ON u.fk_id_device = d.id_device
JOIN activity_type at ON s.fk_id_partner = at.fk_id_partner AND s.fk_id_activity = at.id_activity_type
JOIN partners p ON s.fk_id_partner = p.id_partner
GROUP BY
    s.view_ct, s.click_ct, s.promo_ct, e.time_start, e.time_end, e.date_event, u.region,
    u.birth_dt, u.registration_dt, d.platform, d.os_version, d.phone, p.name_partner, s.team;
GO
```

Рис. 24 Уявлення MetricsSummary

1. s.view_ct - кількість переглядів, які користувач здійснив у сесії.
2. s.click_ct - кількість кліків, зроблених користувачем у сесії.
3. s.promo_ct - кількість промо-кліків, зроблених користувачем у сесії.
4. e.time_start - час початку сесії.
5. e.time_end - час завершення сесії.
6. session_time - тривалість сесії в хвилинах, розрахована як різниця між e.time_end і e.time_start.
7. e.date_event - дата, коли подія відбулася.
8. u.region - регіон, з якого користувач прийшов.

9. years_old - вік користувача, розрахований на основі дати народження u.birth_dt.
10. u.registration_dt - дата реєстрації користувача.
11. d.platform - платформа пристрою користувача.
12. d.os_version - версія операційної системи пристрою.
13. d.phone - модель телефону користувача.
14. total_revenue - загальний дохід, розрахований на основі кількість кліків, переглядів, та промо-кліків, помножені на відповідний дохід за тип активності (at.revenue).
15. p.name_partner - назва партнера, з яким пов'язана сесія.
16. s.team - команда, до якої належить сесія.

- [UserActivity] – з назви стає зрозуміло що це уявлення більше фокусується на користувачах. Воно створює вид, що агрегує дані за ідентифікатором користувача (u.id_customer) та типом активності (at.activity_name), розраховуючи при цьому кількість кліків, переглядів, промо-кліків, а також дохід від кожного типу активності. Запит використовує функцію COALESCE для запобігання NULL значенням у випадку відсутності даних.
    1. u.id_customer - ідентифікатор користувача.
    2. activities_count - загальна кількість активностей користувача.
    3. click_ct - кількість кліків користувача по типу пропозиції.
    4. click_revenue - дохід від кліків користувача по типу пропозиції, розрахований за допомогою множення кількості кліків на відповідний дохід (at.revenue).
    5. view_ct - кількість переглядів користувача по типу пропозиції.

6. view_revenue - дохід від переглядів користувача по типу пропозиції, розрахований за допомогою множення кількості переглядів на відповідний дохід (at.revenue).
7. promo_ct - кількість промо-кліків користувача.
8. promo_revenue - дохід від промо-кліків користувача, розрахований за допомогою множення кількості промо-кліків на відповідний дохід (at.revenue).
9. total_revenue - загальний дохід, який є сумою доходів від кліків, переглядів, та промо-кліків.

- [RegionalActivity] – уявлення яке акцентується на локаціях користувачів.
- [PartnerActivity] – уявлення яке акцентуэться на те які активності у яких партнерів є популярними.
- [UserDemographics] – уявлення, у порівнянні з RegionalActivity, більше акцентується на діяльність користувачів у різних локаціях.

Після створення таких видів уявлень робота аналітиків стає простішим, оскільки основні метрики та агреговані дані уже підготовлені для подальшого використання. Аналітики можуть тепер зосередитися на витягуванні корисних висновків та стратегій, не витрачаючи час на ручний збір та обробку даних. Вони можуть легко імпортувати ці уявлення у інструменти для бізнес-аналізу, такі як Power BI, для створення зрозумілих дашбордів, графіків та прогнозів.

Крім того, уявлення дозволяють створити єдиний "істинний джерело" даних, що зменшує ризик появи невідповідностей або конфліктів у даних. Аналітики можуть також легше поділитися цими уявленнями з іншими відділами компанії, що сприяє кращому взаєморозумінню та швидшому прийняттю рішень на всіх рівнях.

Використання уявлень також полегшує процес аудиту даних. Оскільки всі ключові метрики та розрахунки вже зібрані в одному місці, аудиторам легше перевірити достовірність та консистентність даних, не вникаючи в деталі базових таблиць або зложені запити.

Таким чином, створення уявлень значно оптимізує робочий процес аналітиків, дозволяючи їм зосередитися на більш високорівневих задачах аналізу та стратегічного планування.

4.2 Результати використання завдань якості даних

У розділі 3.3 детально розглянулі завдання, які були інтегровані в DAG у Airflow. Деякі з цих завдань спеціально розроблені для контролю якості вхідних даних. Щоб забезпечити високий рівень обізнаності щодо цієї перевірки, система автоматично надсилає повідомлення в Telegram у випадку перевищення встановлених лімітів на некоректні або неконсистентні дані. Оскільки сама система передбачає що дані будуть вантажитися у СД раз на день, то і повідомлення про наявність проблем у даних будуть приходити лише раз на день. Було створено групу та додано бота, який і має надсилати ці попередження

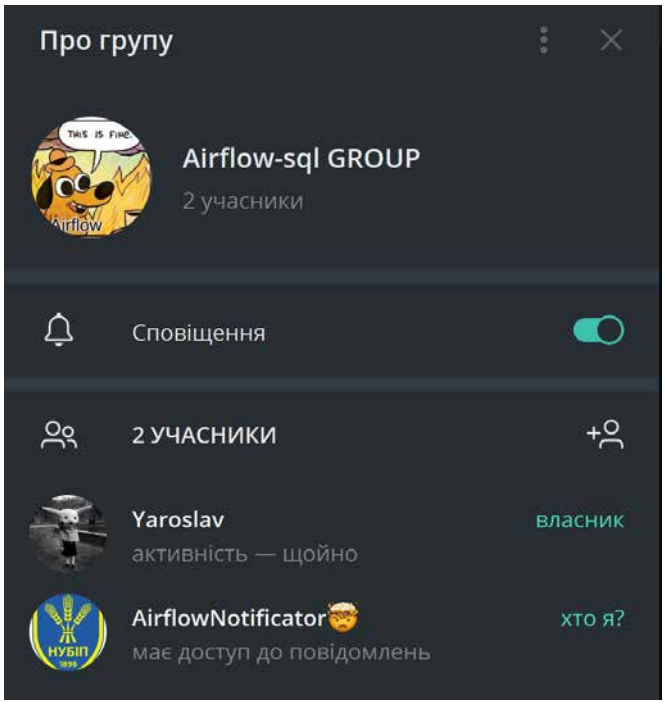


Рис. 25 Група для попереджень

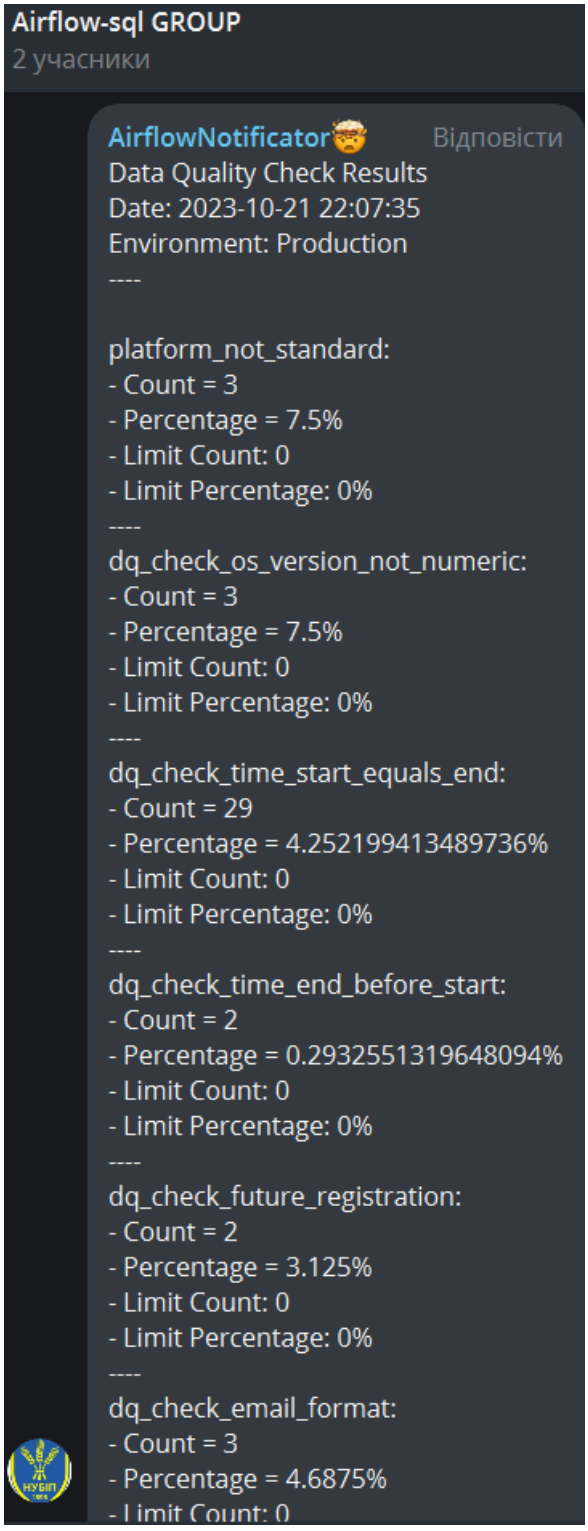


Рис. 26 Повідомлення про помилки у даних відправлено у групу

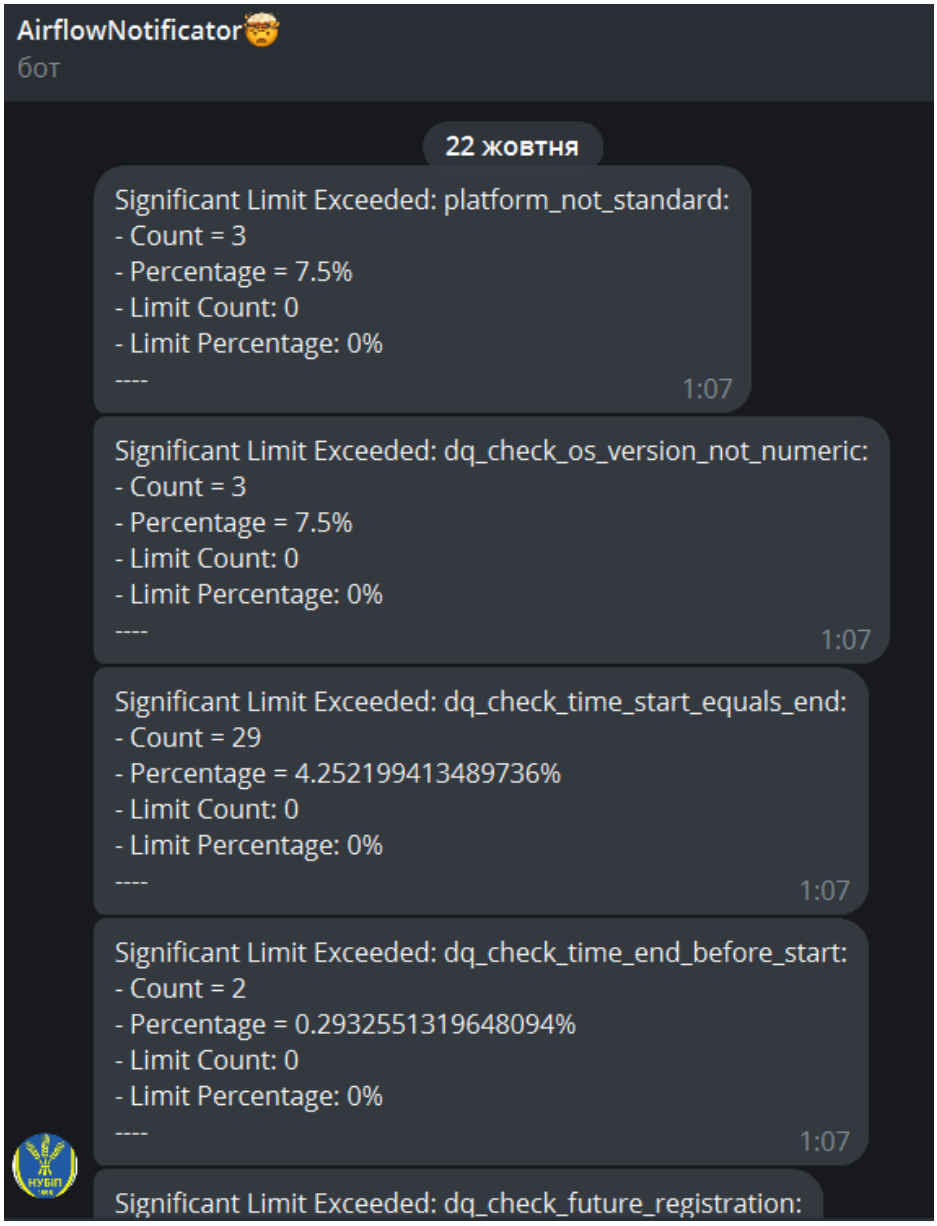


Рис. 27 Повідомлення про помилки у даних відправлено у приватні повідомлення

Як ми бачимо прийшли повідомлення у групу (рис. 25) про те що знайдено помилки по цілому ряду перевірок, вони ж продубльовані у приватні повідомлення зазначених людей (рис. 27) оскільки у два рази перевищуються допустимі ліміти. Суто для наглядності ліміти кожної з помилок були виставлені 0% та 0 рядків допускається з помилками.

Деякі з помилок можна виправити, наприклад помилок можна виправити швидко, наприклад dq_check_time_end_before_start – ця помилка, скоріш за все, спричинена неправильною логікою роботи платформи по збору даних діяльності користувача або її реалізація оскільки ця перевірка робить порівняння початку на кінця часу однієї сесії і якщо в розрізі 1 дня є сесії які мають кінцевий час сесії менший ніж початковий – то це рахується як помилка. Для того щоб виправити цю помилку, найбистріший метод буде просто поміняти значення початку та кінця сесії місцями, та повідомити команду яка реалізувала цю частину збору інформації на фронтенді про те що така проблема було відправлена саме ними.

```
UPDATE [dbo].[event_ts]
SET time_end = time_start,
    time_start = time_end
WHERE date_event = '2023-10-20'
AND time_end < time_start
```

Рис. 28 Приклад запиту для виправлення помилок які викликали dq_check_time_end_before_start

Однак далеко не всі проблеми можливо вирішити вже після збору, у зазначених повідомленнях ми бачимо що емейли для деяких нових користувачів відправлено у неправильному форматі, в цьому випадку на стороні СД ми вже нічого не зможемо зробити. Також бачимо що перевірка dq_check_os_version_not_numeric, яка перевіряє що версія платформи девайсу зазначена в цифрах, має великий відсоток поганих даних.

Щоб розуміти кому адресувати виправлення тих або інших помилок які дата інженери не в змозі вирішити на своєму боку можна скористатися гарною практикою поміткою даних з їх джерелом відправника, тобто в нашому випадку це є фронтенд команди, які реалізували якусь частину збору даних, яка має певні помилки. Для цього створимо уявлення для команд та підрахунку даних, які вони неправильно генерують

```sql
CREATE VIEW [dbo].[TeamIssuesAnalyze] AS
SELECT
    team,
    issue,
    COUNT(*) AS affected_rows
FROM (
    SELECT
        s.team,
        CASE
            WHEN DATEDIFF(MINUTE, e.time_start, e.time_end) = 0 THEN 'session_time = 0'
            WHEN s.click_ct = 50 THEN 'CT anomaly'
            WHEN d.platform NOT IN ('ios', 'android') THEN 'platform not in ios, android'
            WHEN e.time_end < e.time_start THEN 'time_end < time_start'
            WHEN u.registration_dt > GETDATE() THEN 'future registration date'
            WHEN CHARINDEX('@', u.email) = 0 THEN 'incorrect email format'
            ELSE NULL
        END AS issue
    FROM session s
    JOIN event_ts e ON s.fk_id_event_ts = e.id_datetime
    JOIN [user] u ON s.fk_id_user = u.id_customer
    JOIN device d ON u.fk_id_device = d.id_device
    WHERE
        DATEDIFF(MINUTE, e.time_start, e.time_end) = 0 OR
        s.click_ct = 50 OR
        d.platform NOT IN ('iOS', 'Android') OR
        e.time_end < e.time_start OR
        u.registration_dt > GETDATE() OR
        CHARINDEX('@', u.email) = 0
) AS Issues
WHERE issue IS NOT NULL
GROUP BY team, issue;
```

Рис. 29 Запит для уявлення

У запиті, що зображено на рис. 29 зібрані назви команд які є джерелом даних, помилки та кількість рядків.

Кількість строк	Команда	Вплив на усі дані	Помилки
8664	qwerty_prof	4,81%	platform not in ios, android
8483	qwerty_prof	4,71%	incorrect email format
5670	qwerty_prof	3,15%	session_time = 0
3190	qwerty_prof	1,77%	future registration date
799	qwerty_prof	0,44%	time_end < time_start
195	BEST_MOBILE_TEAM	0,11%	platform not in ios, android
174	BEST_MOBILE_TEAM	0,10%	incorrect email format
115	BEST_MOBILE_TEAM	0,06%	session_time = 0
76	BEST_MOBILE_TEAM	0,04%	future registration date
15	BEST_MOBILE_TEAM	0,01%	time_end < time_start
10	TEAM_A	0,01%	incorrect email format
8	TEAM_A	0,00%	session_time = 0
7	TEAM_A	0,00%	platform not in ios, android
5	TEAM_A	0,00%	future registration date
1	TEAM_A	0,00%	time_end < time_start
27412		**15,23%**	

Рис. 30 Результуюча таблиця

Рис. 30 демострує створену таблицю з назвами команд та тим які помилки вони надсилають, як бачимо 3 команди, а саме qwerty_prof, BEST_MOBILE_TEAM та TEAM_A сумарно надіслали таку кількість помилок, що 15 відсотків загального об’єму даних містять неякісні дані.

Найкращим шляхом вирішенням цих помилок буде направлення роз’яснення по кожній помилці кожній з команд та консультація з представниками Segment щодо вирішення цих проблем в подальшому.

Оскільки бізнес має приносити дохід то особливу увагу потрібно приділяти саме даним які містять фінансові елементи, тому додатково досліджено та в подальшому буде впроваджено завдання на перевірку саме таких елементів даних.

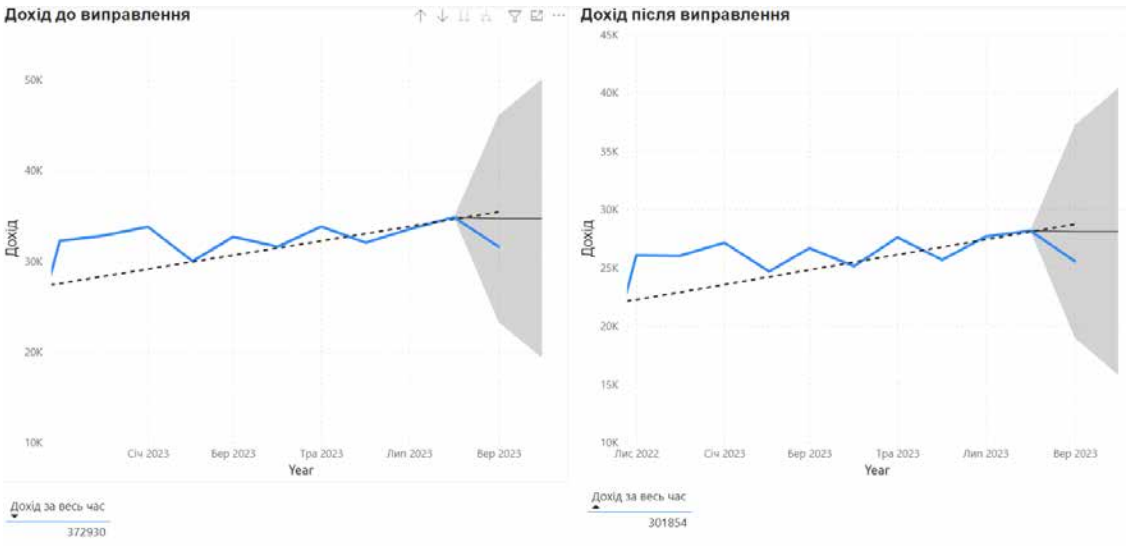


Рис. 31 Щомісячний дохід та прогноз

На рис. 31 представлено порівняння даних до та після видалення аномалій у фінансових показниках. Різниця у майже 70 тисяч є дуже суттєвою для компанії з середніми доходами приблизно 30 тисяч на місяць. Ці аномалії проявляються у видачі надмірно високих значень активності користувачів. Для ідентифікації цих аномалій використовувалася проста машинна модель, яка розраховує середні значення показників для різних стовпців. Наприклад, якщо ми розглядаємо переходи за промо-посиланнями, то середнє значення може становити 3 переходи, тоді як максимальне значення з фільтрованої вибірки (де виключено переходи на рівні 15 і вище) може досягати 5 переходів. Така ситуація вимагає додаткової уваги та аналізу. Для простого виявлення аномалій можна використовувати метод заснований на середньому значенні та стандартному відхиленні. Формула для виявлення аномалій у даних може бути такою:

$$\text{Аномалія}=\{x \mid x>\mu+k\sigma \text{ або } x<\mu-k\sigma\} \quad (4.1)$$

де μ — це середнє значення вибірки;

σ — стандартне відхилення;

k — константа, яка визначає "силу" фільтрації (зазвичай вибирається в діапазоні від 2 до 3).

1. Розрахувати середнє значення (μ) та стандартне відхилення (σ) для даних зі стовпця (наприклад, "total_revenue").
2. Вибрати значення k, зазвичай 2 або 3.
3. Виявити аномалії, використовуючи формулу 4.1.

Ця проста модель може бути реалізована в Python за допомогою бібліотеки Pandas або NumPy для обрахунку статистичних метрик та фільтрації даних.

Ще яке приклад логічної помилки: дата реєстрації яка має валідний формат і наче відправляється з фронтенду бувають з майбутньою датою

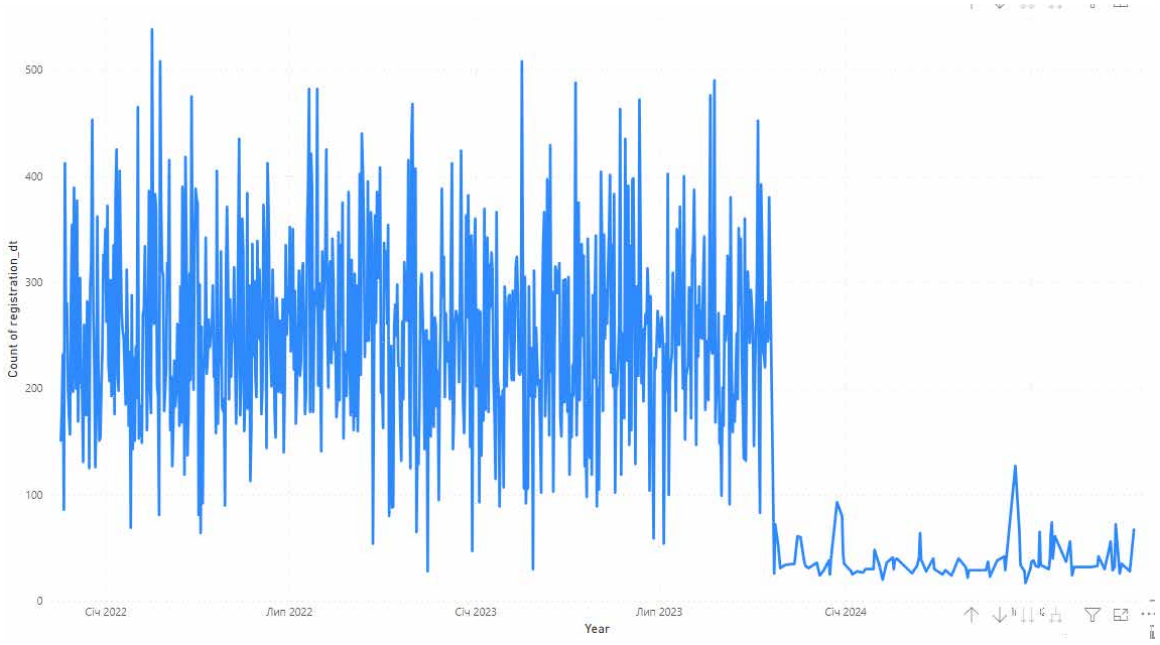


Рис. 32 Графік реєстрацій нових користувачів (на момент 21 жовтня 2023)

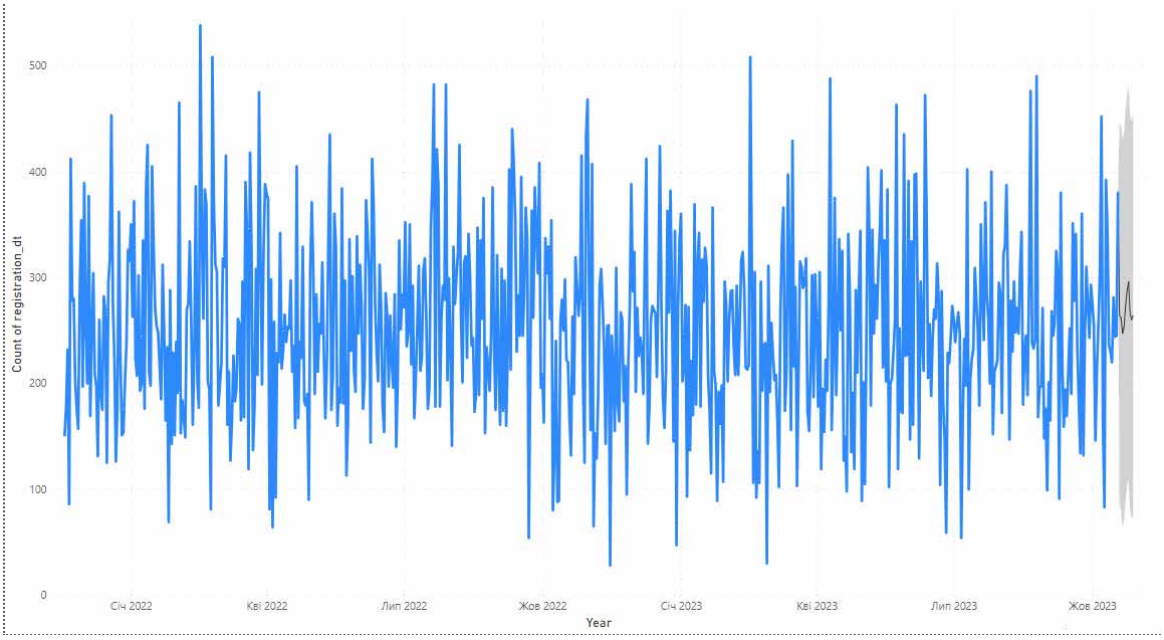


Рис. 33 Графік реєстрацій нових користувачів (на момент 21 жовтня 2023) з виключеними даними за майбутні дати

Як ми бачимо по графіку на рис. 32 то для повноцінного аналізу та прогнозу доведеться виключати дані за майбутні реєстрації та втрачати безцінні дані користувачів (на рис. 33 бачимо вже очищені дані).

Також було проведену перевірку тривалості сесій користувачів.

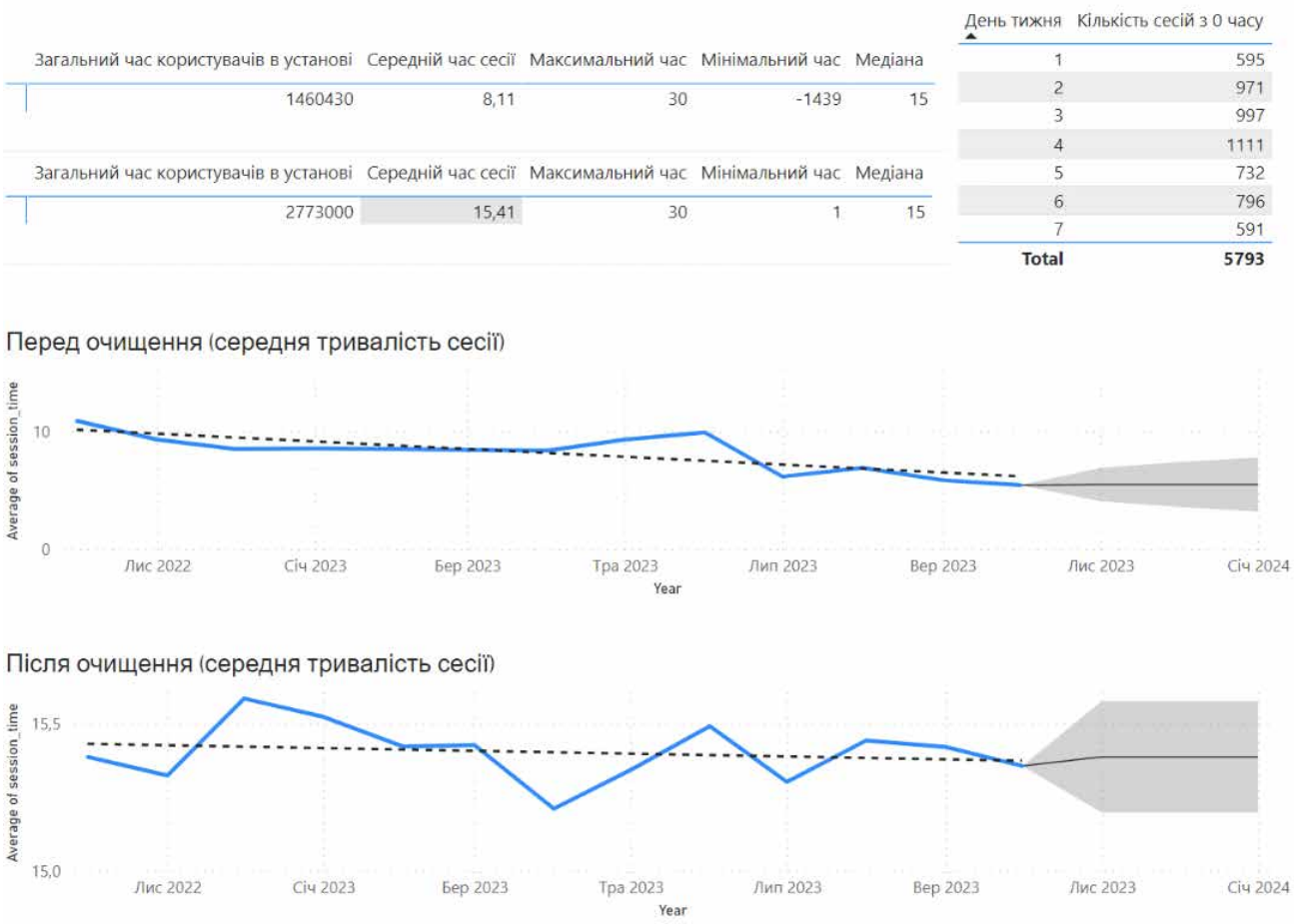


Рис. 34 Порівняння аналізу даних тривалості сесій

Аналіз (див. рис. 34) даних тривалості сесії виявив аномальний мінімальний час для сесії (від'ємне значення), що було призведено невірним розрахунком часу (початок сесії день 1 – кінець сесії день 2) – до такого висновку можна прийти порівнявши мінімальне значення тривалості сесії до очищення, оскільки 1 доба = 1440 хвилин (значення -1439 буде для сесії з початком у 23:59:00 та закінченням у 00:00 наступного дня), отже щоб очистити даних від цієї помилки, потрібно було просто брати до уваги не тільки години, але й день коли сесія почалася та день коли вона закінчилася. На рис. 34 у порівнянні показані як виглядають аналізи тривалості сесії після очищення їх.

4.3 Результати аналізу та прогнозу

Найголовнішим аналізом для усіх типів установ завжди був дохід, тому і почати потрібно з нього в першу чергу.

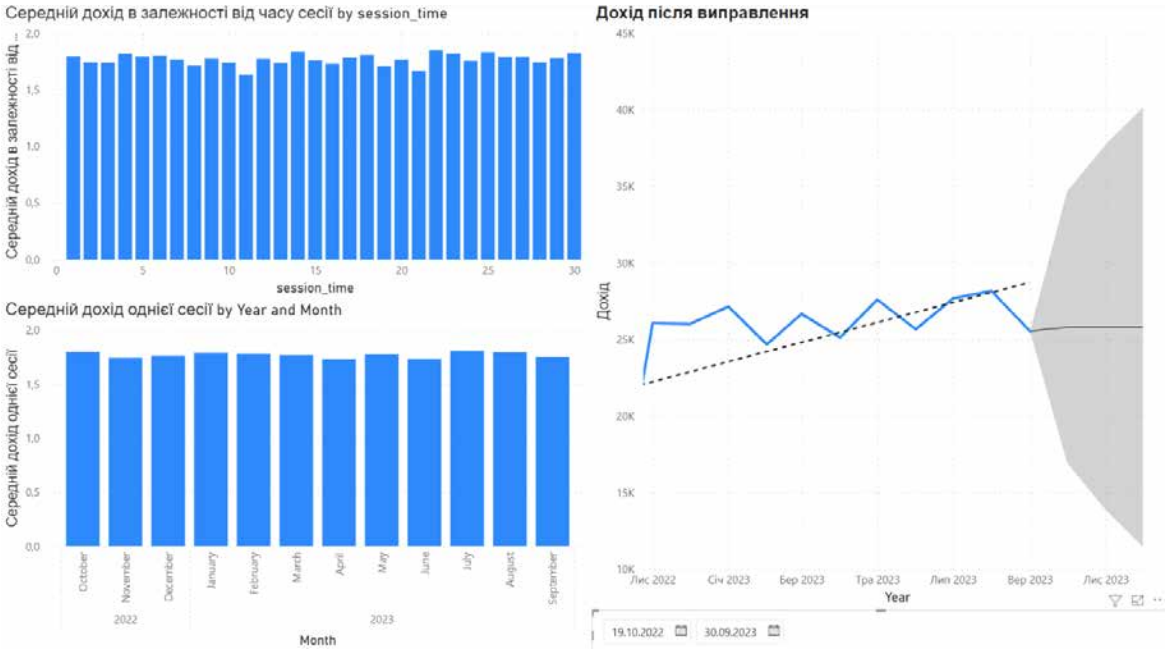


Рис. 35 Аналіз доходу

На дашборді з рис.35 можна побачити 3 графіка: середній дохід однієї сесії в залежності від часу сесії, середній дохід однієї сесії по місяця та графік щомісячного прибутку з прогнозом, варто зазначити, що дані після їх виправлення значно змінилися та тепер відповідають дійсності.

Бачимо що з першого графіку сильно вирізняється лише сесії 11 хвилин які приносять лише 1.63, тому аналітики продовжать наголошувати на тому, що користувачі повинні залишатися на сайті якомога більше задля отримання найбільшого прибутку від них.

На другому графіку вже менш помітно залежність між прибутком та місяцем року.

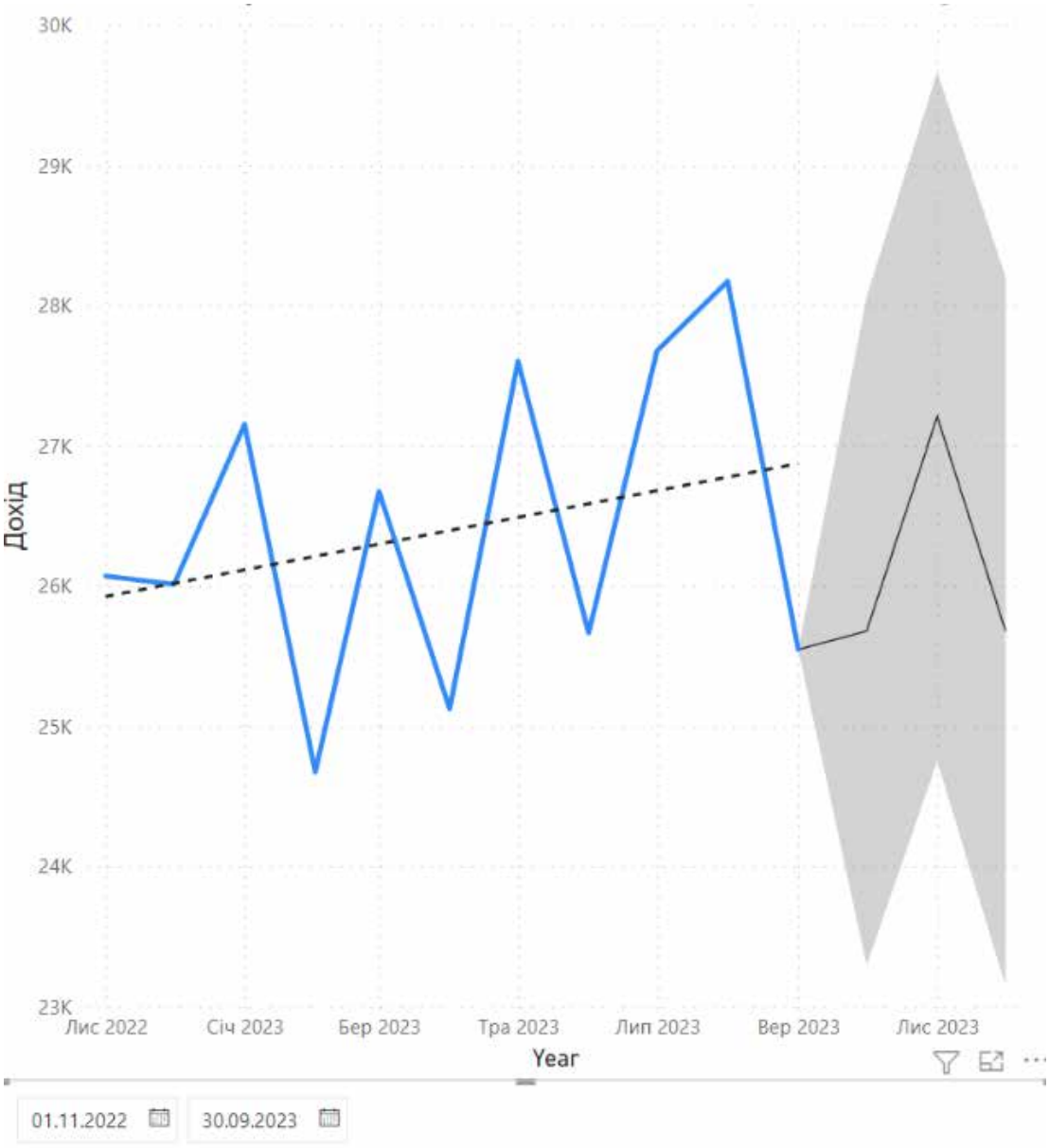
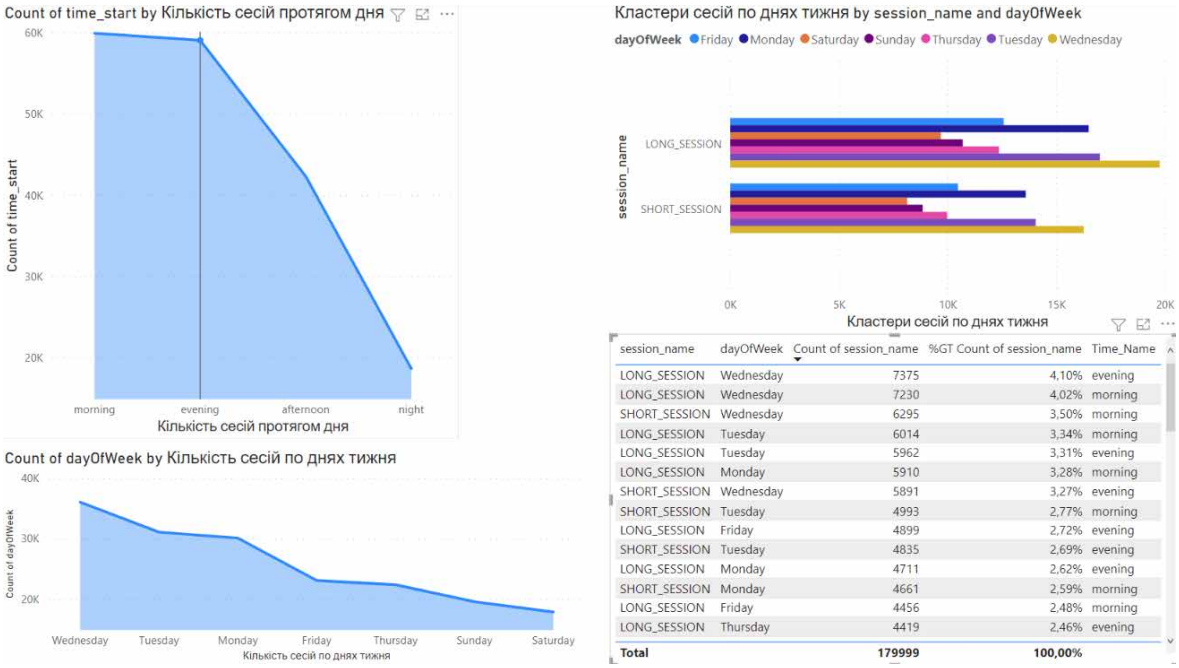


Рис. 36 Графік доходу з листопад 2022 по кінець вересня 2023

На третьому графіку бачимо що лінія тренду (штрихова) показує що є зріст, однак у системі дані не за цілий місяць жовтня 2022, тому з вибірки приберемо з вибірки цей місяць та побачимо (див. рис.36) що навіть так тренд залишається та з середнього значення 26 тисяч за майже рік установа прийшла до майже 27 тисяч. Прогнозування, яке виконане у середовищі Power BI, показує що через те що дохід дуже коливається то прогнозування не є дуже точним. У випадку цього графіку має сенс продовжувати розроблювати підходи до

розробки компанії реклами та стратегії компанії так само, як це було влітку 2023, оскільки саме в цей час помічено найбільший ріст протягом 3 місяців.



session_name	dayOfWeek	Count of session_name	%GT Count of session_name	Time_Name
LONG_SESSION	Wednesday	7375	4,10%	evening
LONG_SESSION	Wednesday	7230	4,02%	morning
SHORT_SESSION	Wednesday	6295	3,50%	morning
LONG_SESSION	Tuesday	6014	3,34%	morning
LONG_SESSION	Tuesday	5962	3,31%	evening
LONG_SESSION	Monday	5910	3,28%	morning
SHORT_SESSION	Wednesday	5891	3,27%	evening
SHORT_SESSION	Tuesday	4993	2,77%	morning
LONG_SESSION	Friday	4899	2,72%	evening
SHORT_SESSION	Tuesday	4835	2,69%	evening
LONG_SESSION	Monday	4711	2,62%	evening
SHORT_SESSION	Monday	4661	2,59%	morning
LONG_SESSION	Friday	4456	2,48%	morning
LONG_SESSION	Thursday	4419	2,46%	evening
Total		**179999**	**100,00%**	

Рис. 37 Дашборд кількості сесій

На наступному дашборді з рис. 37 зображено 3 графи та 1 таблиця дистрибуції: кількість сесій по часовим показникам дня, кількість сесій по днях тижня, кластери типів сесій по днях тижня та таблиця типів сесії з їх дистриб’юцією по днях.

Перший графік показує що найбільша кількість сесій приходиться саме на ранок (з 6 до 12) може свідчити про кілька різних факторів. Спочатку, це може бути зв'язано з поведінкою користувачів: можливо, це час, коли вони найактивніші в інтернеті, перш ніж почати робочий день або зайнятися іншими справами. Також це може бути результатом географічного розподілу користувачів, якщо, наприклад, більшість з них знаходяться в певному часовому поясі. Також це дає змогу використовувати цей часовий проміжок для проведення A/B тестів, оскільки він буде найрепрезентативнішим для користувачів.

Другий графік показує що середа має найбільшу кількість сесій у середу.

Середа часто вважається "піком" робочого тижня, коли багато людей відчувають максимальну продуктивність. Можливо, це час, коли користувачі шукають певні послуги або продукти для вирішення їх поточних задач. Оскільки найбільше сесій у середу зранку можна допустити, що користувачі заходять до або під час початку свого робочого дня.

Розглянемо останній графік та таблицю дистриб’юції. На графіку видно, що кластери майже не мають різниці у дистриб’юції по днях тижня. А у таблиці побачимо, що середа частіше за всього з’являється зранку та є довгою, тобто більше 15 хвилин. З огляду на останній графік та таблицю дистриб'юції, ми можемо зробити кілька важливих спостережень та рекомендацій для компанії. По-перше, незначна різниця в дистриб'юції кластерів по днях тижня може свідчити про стабільну поведінку користувачів, незалежно від конкретного дня. Це може бути важливим фактором для планування маркетингових кампаній, а також для розподілу ресурсів сервера.

По-друге, звертає на себе увагу той факт, що середа ввечері часто характеризується більш тривалими сесіями. Це може свідчити про збільшену відданість користувачів в цей часовий проміжок, і, можливо, їх більшу готовність до взаємодії з пропозиціями.

Наступний дашборд відображає кореляцію між прибутком та діями користувача (кліками, переглядами та переходами)

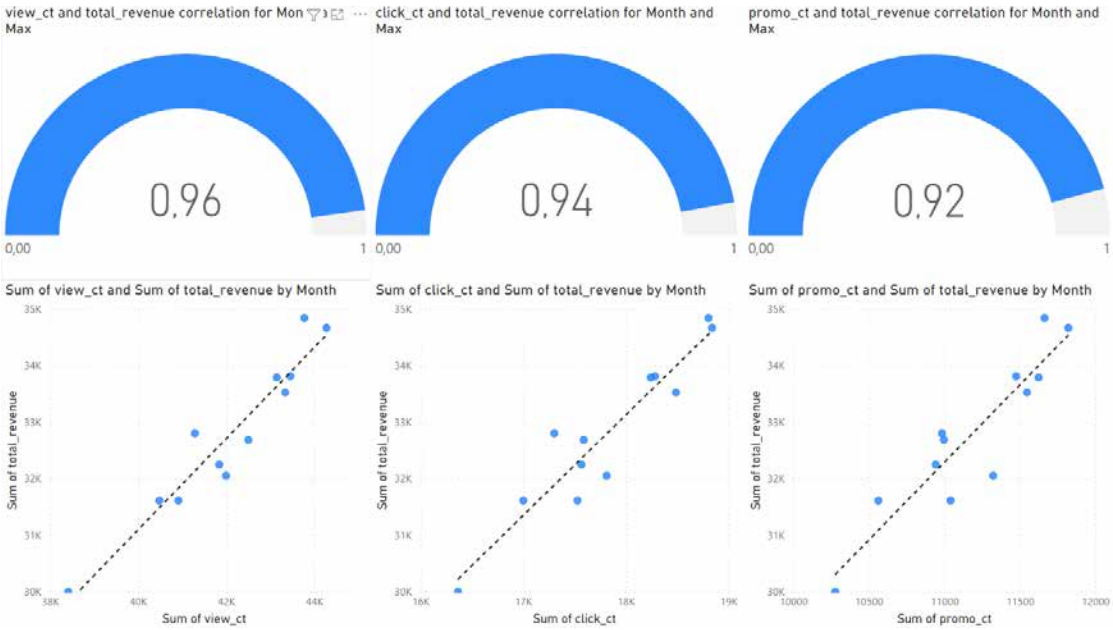


Рис. 38 Дашборд з кореляціми по активностям користувачів

В аналітиці, кореляція є статистичним показником, який оцінює ступінь взаємозв'язку між двома або більше змінними. Значення кореляції варіюється від -1 до +1: значення близьке до +1 свідчить про сильну пряму кореляцію, -1 вказує на сильну обернену кореляцію, тоді як значення близьке до 0 вказує на відсутність кореляції. Важливо розуміти, що кореляція не означає причинно-наслідковий зв'язок. За допомогою аналізу кореляції можна виявляти закономірності в даних та формулювати гіпотези для подальшого дослідження. На рис. 38 висока кореляція між переглядами та загальним доходом, яка дорівнює 0.96, свідчить про сильний прямий взаємозв'язок між цими двома змінними. Це може означати, що збільшення кількості переглядів, як правило, супроводжується збільшенням загального доходу. Однак слід пам'ятати, що кореляція не доказує причинно-наслідкового зв'язку, тому це може бути лише одним з факторів, які впливають на доход.

Що стосується кореляції між переходами та загальними доходами, яка становить 0.92, вона також є високою, але трохи нижчою в порівнянні з першою парою змінних. Це може свідчити про те, що переходи також є ефективними в генерації доходу, але можливо не настільки ефективними, як просто перегляди.

З такою інформацією, аналітики можуть порекомендувати зосередитись на стратегіях, які збільшують кількість переглядів, оскільки це може мати найбільший позитивний вплив на загальний доход. Також може бути корисним подивитись, чи можна оптимізувати переходи для досягнення ще вищих показників доходу.

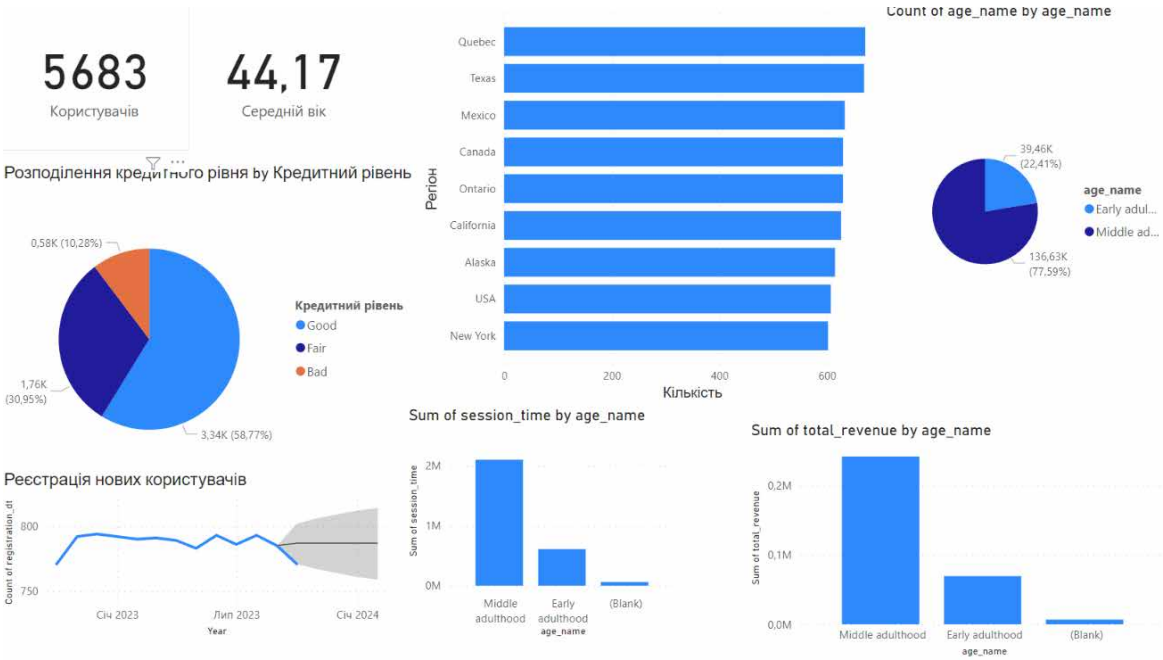


Рис. 39 Дашборд аналітики по користуваачам

Дашборд даних користувачів (див. рис. 39) містить графіки та цифри щодо користувачів системи, з інформацію про яких можна будувати плани та стратегії щодо впровадження тих або інших акцій.

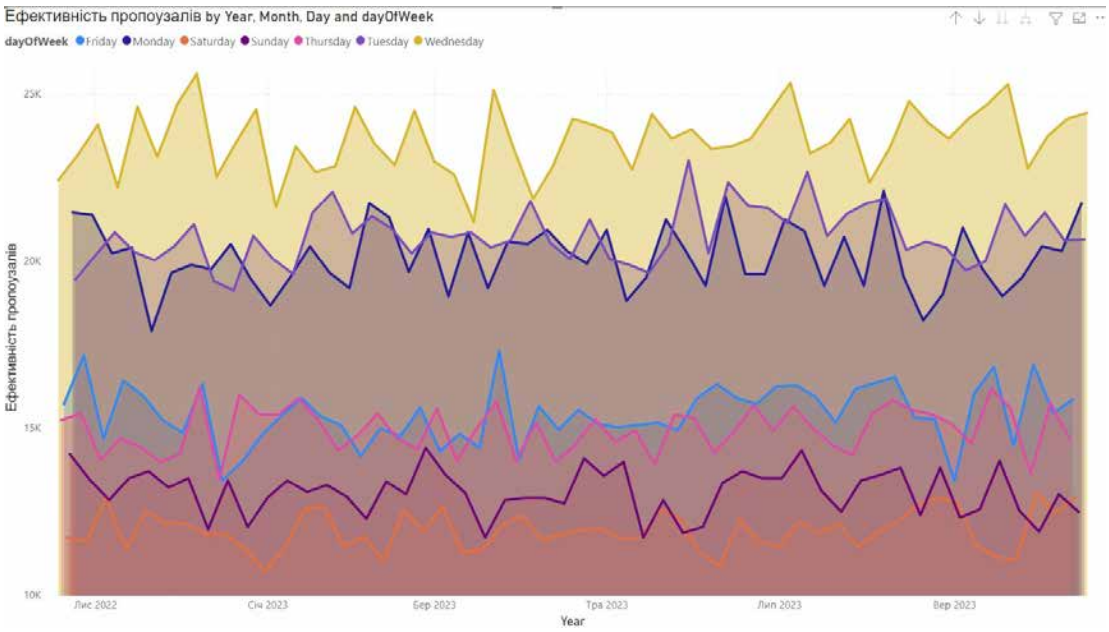


Рис. 40 CTR по днях тижня

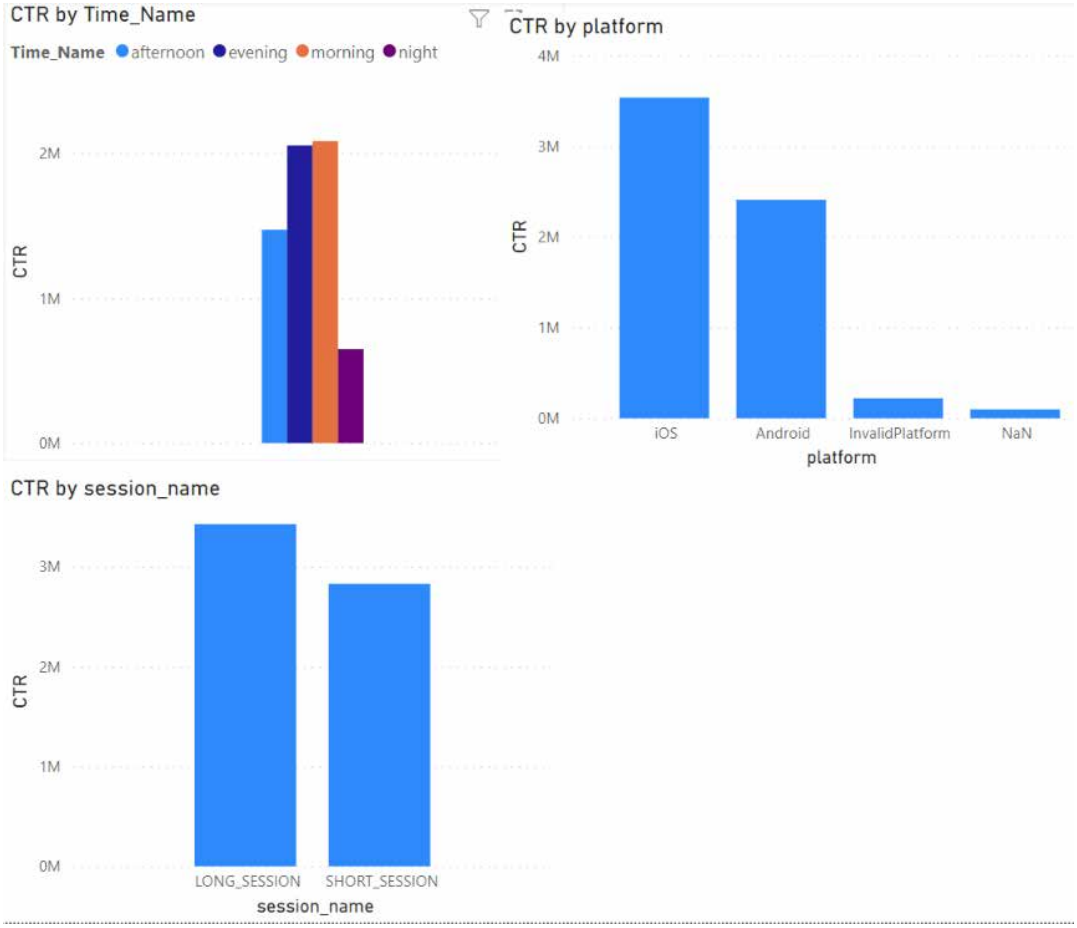


Рис. 41 CTR по різним кластерам

CTR (Click-Through Rate) або коефіцієнт переходу — це метрика в аналітиці, яка вимірює ефективність рекламних кампаній, банерів, посилань тощо. Вона показує відсоток користувачів, які клацнули на рекламний матеріал (в нашому випадку пропозицій партнерів) відносно загальної кількості переглядів цього матеріалу.

CTR = (кількість кліків/кількість переглядів) * 100 (4.2)

Високий CTR може свідчити про те, що рекламний контент або посилання є ефективним і привабливим для аудиторії, тоді як низький CTR може вказувати на неефективність рекламного матеріалу або невідповідність цільовій аудиторії.

Рис. 40 відображає CTR для днів тижня, де лідером залишається бути середа по найвищим показникам, а на рис. 41 різні категорії даних порівнюються між собою у кількості CTR:

- Зранку та ввечері мають найбільші показники набільший
- Девайси на iOS мають найбільший показник
- Довгі сесії мають цей показник кращим

Даний розділ аналізу та прогнозу розкриває ключові метрики доходу, поведінки користувачів та їх взаємозв'язок. Основний фокус ставиться на доход, який є центральним показником ефективності. Дашборди візуалізують тренди доходу, кількість сесій та кореляції між активностями користувачів і доходами. Загалом, дані та їх аналіз надають цінну інформацію для оптимізації маркетингових стратегій, планування ресурсів та покращення користувацького досвіду.

ВИСНОВОК

В ході даного проекту було розроблено систему для збору, аналізу та візуалізації даних користувачів з акцентом на якість даних. Система включає в себе зчитування івентів користувача через Segment, завантаження даних в оперативну СД, перевірку та трансформацію даних за допомогою Apache Airflow, а також подальший аналіз та візуалізацію в Power BI.

Важливим аспектом розробленої системи є контроль якості даних на кожному етапі їх життєвого циклу — від моменту збору до вивантаження в СД. Це особливо критично, оскільки початкові перевірки на фронтенді були виконані не на достатньому рівні якості, що призвело до необхідності виправлення даних на наступних етапах.Хибні або неповні дані можуть суттєво спотворити результати аналітичних досліджень, призвести до невірних прогнозів і, як наслідок, неправильних стратегічних рішень на рівні компанії. Тому висока якість даних не просто підвищує ефективність системи, але і є гарантом точності аналітичних висновків та подальшого успішного розвитку бізнесу.

Технологічний стек включає Python для генерації та перевірки даних, Apache Airflow для оркестрації задач та керуванням даними, а також SQL Server як основне сховище даних. Apache Airflow було розгорнуто в середині Docker-контейнера для спрощення розгортання та забезпечення ізоляції. Всередині Airflow, використовуються різні типи воркерів: webserver, worker, trigger, scheduler, init, postres, та redis для забезпечення різноманітних аспектів роботи системи.

Однією з ключових особливостей системи є висока увага до якості даних. Airflow задіяний для перевірки якості даних перед їх вивантаженням в СД. У випадку виявлення помилок або аномалій, система відправляє нотифікації в Telegram через його API. Вся конфіденційна інформація, така як рядки

підключення до баз даних, зберігається в зашифрованому вигляді в розділі Variables у Airflow.

В результаті, система не тільки забезпечує збір та аналіз великих обсягів даних, але і робить це з високим рівнем надійності та якості. Це, у свою чергу, дозволяє аналітикам в компанії ефективно використовувати ці дані для глибокого аналізу користувачів, їхньої поведінки, та взаємодії з сервісами. Такий підхід значно підвищує цінність аналітичної системи для бізнесу.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Brown, B., Chui, M., & Manyika, J. (2011). Are you ready for the era of ‘big data’?
2. Forrester Research. (2020). The Forrester Wave: Financial Risk Management Platforms. Forrester Consulting
3. Krug, Steve. (2000) "Don't Make Me Think: A Common Sense Approach to Web Usability."
4. Object Management Group. (2017). About the Unified Modeling Language Specification Version 2.5. Object Management Group
5. Kulak, D., & Guiney, D. (2000). Use Cases: Requirements in Context. ACM SIGCHI Curriculum Development Group.
6. Martinez, L. (2020). "Apache Airflow in Data Engineering", Data Science Review
7. Voigt, P., & von dem Bussche, A. (2017). The EU General Data Protection Regulation (GDPR). Cham: Springer International Publishing
8. Maxime Beauchemin, (2015). "Airflow: a platform to programmatically author, schedule, and monitor workflows"
9. Email Marketing Law and Google’s New Email Delivery Restrictions [Електронний ресурс] – Режим доступу до ресурсу: https://www.lexology.com/library/detail.aspx?g=051787d9-64a3-4a3e-87b4-490435387bf4
10. OS Market Share – Mobile – United States of America [Електронний ресурс] – Режим доступу до ресурсу: https://gs.statcounter.com/os-market-share/mobile/united-states-of-america

11. Документація Apache Airflow [Електронний ресурс] – Режим доступу до ресурсу: https://airflow.apache.org/docs/apache-airflow/stable/index.html