Державний торговельно-економічний університет
Кафедра інженерії програмногозабезпечення та кібербезпеки

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Автоматизована система збору, аналізу та зберігання маркетингових даних з соціальних мереж»

Студента 2м курсу, 2 групи, спеціальності 121 «Інженерія програмного забезпечення» освітньої програми «Інженерія програмного забезпечення» _____ підпис студента — Голуб Юрій Олексійович

Науковий керівник кандидат педагогічних наук, доцент кафедри інженерії програмного забезпечення та кібербезпеки _____ підпис керівника — Жирова Тетяна Олександрівна

Гарант освітньої програми кандидат педагогічних наук, доцент кафедри інженерії програмного забезпечення та кібербезпеки _____ підпис гаранта — Котенко Наталія Олексіївна

КИЇВ – 2023

Факультет інформаційних технологій
Кафедра інженерії програмного забезпечення та кібербезпеки
Освітній ступінь магістр
Освітня програма 121 «Інженерія програмного забезпечення»

Затверджую
Зав. кафедри інженерії програмного забезпечення та кібербезпеки
Криворучко О. В.
«13» грудня 2023 р.

Завдання
на випускну кваліфікаційну роботу студентові
Голубу Юрію Олексійовичу
(прізвище, ім'я, по батькові)

1. Тема випускної кваліфікаційної роботи «*Автоматизована система збору, аналізу та зберігання маркетингових даних з соціальних мереж*»
2. Затверджена наказом ректора від «06»*грудня 2022 р. № 3285*
2. Строк здачі студентом закінченої роботи 29 листопада 2023
3. Цільова установка та вихідні дані до роботи

Мета роботи: розробка алгоритмів, методів та засобів для автоматизованої генерації, збору маркетингових даних для подальшого аналізу
Об'єкт дослідження: процеси автоматизованої система збору, аналізу та зберігання маркетингових даних з соціальних мереж.
Предмет дослідження: компоненти, які входять до складу автоматизованої системи, спрямованої на збір, аналіз та зберігання маркетингових даних з різних соціальних мереж. Також включає в себе алгоритми та методи, використовувані для здійснення збору, обробки та аналізу даних з метою подальшого використання їх у маркетингових стратегіях та прийнятті рішень.

4. Консультанти роботи із зазначенням розділів, які консультують:

Розділ	Консультант (прізвище, ініціали)	Підпис, дата	
		Завдання видав	Завдання прийняв

5. Зміст випускної кваліфікаційної роботи (перелік питань за кожним розділом)

ВСТУП

РОЗДІЛ 1. ТЕОРЕТИЧНЕ ДОСЛІДЖЕННЯ МЕХАНІЗМІВ АВТОМАТИЗОВАНОЇ СИСТЕМИ ЗБОРУ, АНАЛІЗУ ТА ЗБЕРІГАННЯ МАРКЕТИНГОВИХ ДАНИХ.

1.1. Поняття та теоретичні основи збору маркетингових даних з соціальних мереж.

1.2. Аналіз відомих методів систем збору, аналізу та зберігання даних

1.3. Характеристика існуючих методів та програмних засобів збору аналізу та зберігання маркетингових даних

1.4 Висновки до першого розділу

РОЗДІЛ 2. ПРОЕКТУВАННЯ СИСТЕМИ ЗБОРУ, АНАЛІЗУ ТА ЗБЕРІГАННЯ МАРКЕТИНГОВИХ ДАНИХ.

2.1. Постановка задачі

2.2. Метод ETL або ELT пайплайнів

2.3. Метод MPP (Massively Parallel Processing)

2.4. OLTP та OLAP методи та системи БД

2.5 Висновки до другого розділу

РОЗДІЛ 3. Розробка програмного забезпечення системи збору, аналізу та зберігання маркетингових даних з соціальних мереж

3.1. Архітектура системи збору, аналізу та зберігання маркетингових даних з соціальних мереж

3.2. Програмна реалізація системи клієнтських даних для авторизації маркетингових даних з соціальних мереж

3.3. Програмна реалізація системи збору, аналізу та зберігання маркетингових даних з соціальних мереж

3.4. CloudFunction

3.5 Висновки до третього розділу

ВИСНОВКИ ТА ПРОПОЗИЦІЇ

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

ДОДАТКИ

6. Календарний план виконання роботи

№ пор.	Назва етапів випускної кваліфікаційної роботи	Строк виконання етапів роботи	
		за планом	фактично
1	2	3	4
1.	*Вибір теми випускної кваліфікаційної роботи*	*07.11.2022*	*07.11.2022*
2.	*Розробка та затвердження завдання на роботу магістра (стац/заоч)*	*13.12.2022*	*13.12.2022*
3.	*Вступ та перелік літературних джерел*	*24.02.2023*	*24.02.2023*
4.	*Розробка технічного завдання*	*15.03.2023*	*15.03.2023*
5.	*Розділ 1. Теоретичне дослідження механізмів автоматизованої системи збору, аналізу та зберігання маркетингових даних*	*10.04.2023*	*10.04.2023*
6.	*Розділ 2. Проектування системи збору, аналізу та зберігання маркетингових даних.*	*24.05.2023*	*24.05.2023*
7.	*Розділ 3. Розробка програмного забезпечення системи збору, аналізу та зберігання маркетингових даних з соціальних мереж.*	*06.09.2023*	*06.09.2023*
8.	*Розробка програми та методики тестування*	*18.10.2023*	*18.10.2023*
9.	*Написання наукової статті*	*17.05.2023*	*17.05.2023*
10.	*Керівництво користувача*	*25.10.2023*	*25.10.2023*
11.	*Висновки та пропозиції*	*01.11.2023*	*01.11.2023*
12.	*Здача випускної кваліфікаційної роботи на кафедру (перша перевірка)*	*06.11.2023*	*06.11.2023*
13.	*Підготовка автореферату та презентації доповіді*	*06.11.2023*	*06.11.2023*
14.	*Попередній захист випускної кваліфікаційної роботи*	*20.11.2023 – 24.11.2023*	*20.11.2023 – 24.11.2023*
15.	*Здача зброшурованої випускної кваліфікаційної роботи*	*01.12.2023*	*01.12.2023*
16.	*Зовнішнє рецензування випускної кваліфікаційної роботи*	*02.12.2023*	*02.12.2023*
17.	*Підготовка до публічного захисту випускної кваліфікаційної роботи*	*05.12.2023-06.12.2023*	*05.12.2023-06.12.2023*

7. Дата видачі завдання «13» грудня 2022 р.

8. Науковий керівник випускної кваліфікаційної роботи Жирова Т.О.
(прізвище, ініціали, підпис)

9. Гарант освітньої програми Котенко Н.О.
(прізвище, ініціали, підпис)

10. Завдання прийняв до виконання студент Голуб Ю.О.
(прізвище, ініціали, підпис)

11. Відгук керівника випускного кваліфікаційного проєкту

Науковий керівник випускної кваліфікаційної роботи

_____

(підпис, дата)

Відмітка про попередній захист _____

(ПІБ, підпис, дата)

12. Висновок про випускну кваліфікаційну роботу

Випускна кваліфікаційна робота студента _____Голуб Ю.О._____

(прізвище, ініціали)

може бути допущена до захисту екзаменаційній комісії.

Гарант освітньої програми _____Котенко Н.О._____

(прізвище, ініціали, підпис)

Завідувач кафедри _____Криворучко О. В._____

(підпис, прізвище, ініціали)

«_____»_____20 ___ р.

АНОТАЦІЯ

Дана дипломна робота присвячена розробці автоматизованої системи генерації тестових завдань, Дана дипломна робота націлена на розробку майбутньої автоматизованої системи збору, аналізу та зберігання маркетингових даних з соціальних мереж з використанням Python, GCP, Docker, SQL та PostgreSQL. Метою цієї роботи є створення потужного та інноваційного інструменту, який дозволить маркетологам та компаніям ефективно збирати та аналізувати маркетингові дані з різних соціальних мереж з метою покращення стратегій маркетингу. Ключова мета роботи є розробка автоматизованої системи, яка забезпечуватиме автоматичний процес збору маркетингових даних з соціальних мереж, аналізу та зберігання цих даних в структурованому форматі бази даних. Одна з цілей роботи полягає у створенні модуля, що забезпечуватиме зручний та ефективний збір даних з різних соціальних мереж. Для досягнення цієї мети використовувались передові технології програмування, зокрема мова Python, яка є потужним інструментом для обробки та аналізу даних. Крім того, використовувалась платформа Google Cloud Platform (GCP) для забезпечення масштабованості та надійності системи, а також технологія контейнеризації Docker для спрощення розгортання та управління додатком. Для зберігання та організації даних використовувалась база даних SQL з використанням системи управління базами даних PostgreSQL.

Ключові слова: Google Cloud Platform, Python, модульна архітектура, PostgreSQL, SQL, Docker, маркетинг, автоматизована система.

ABSTRACT

This thesis is dedicated to the development of an automated system for generating test tasks. The thesis aims to develop a future automated system for collecting, analyzing, and storing marketing data from social media using Python, GCP, Docker, SQL, and PostgreSQL. The goal of this project is to create a powerful and innovative tool that enables marketers and companies to efficiently gather and analyze marketing data from various social media platforms, with the aim of improving marketing strategies. One of the key objectives of this work is to develop an automated system that facilitates the automatic process of collecting marketing data from social media, analyzing it, and storing the data in a structured database format. One of the project's goals is to create a module that ensures convenient and efficient data collection from different social media platforms. To achieve this goal, state-of-the-art programming technologies, particularly Python, which is a powerful tool for data processing and analysis, were utilized. Additionally, the Google Cloud Platform (GCP) was employed to ensure scalability and system reliability, along with Docker containerization technology to simplify deployment and application management. For data storage and organization, a SQL database using the PostgreSQL database management system was utilized.

Keywords: Google Cloud Platform, Python, modular architecture, PostgreSQL, SQL, Docker, marketing, automated system.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

GCP– Google Cloud Platform

API –Application Programming Interface

ЗМІСТ

ВСТУП 3

РОЗДІЛ 1. ТЕОРЕТИЧНЕ ДОСЛІДЖЕННЯ МЕХАНІЗМІВ АВТОМАТИЗОВАНОЇ СИСТЕМИ ЗБОРУ, АНАЛІЗУ ТА ЗБЕРІГАННЯ МАРКЕТИНГОВИХ ДАНИХ. 7

1.1. Поняття та теоретичні основи збору маркетингових даних з соціальних мереж 7

1.2. Аналіз відомих методів систем збору, аналізу та зберігання даних 16

1.3. Характеристика існуючих методів та програмних засобів збору аналізу та зберігання маркетингових даних. 19

1.4. Висновки до розділу 1 23

РОЗДІЛ 2. ПРОЕКТУВАННЯ СИСТЕМИ ЗБОРУ, АНАЛІЗУ ТА ЗБЕРІГАННЯ МАРКЕТИНГОВИХ ДАНИХ 25

2.1. Постановка задачі 25

2.2. Метод ETL або ELT пайплайнів 27

2.3. Метод MPP (Massively Parallel Processing) 31

2.4. OLTP та OLAP методи та системи БД 33

2.5. Висновки до розділу 2 39

РОЗДІЛ 3. РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ СИСТЕМИ ЗБОРУ, АНАЛІЗУ ТА ЗБЕРІГАННЯ МАРКЕТИНГОВИХ ДАНИХ З СОЦІАЛЬНИХ МЕРЕЖ 41

3.1. Архітектура системи збору, аналізу та зберігання маркетингових даних з соціальних мереж 41

3.2. Програмна реалізація системи клієнтських даних для авторизації маркетингових даних з соціальних мереж 42

3.3. Програмна реалізація системи збору, аналізу та зберігання маркетингових даних з соціальних мереж 45

3.4. CloudFunction 52

3.5 Висновки до розділу 3 56

ВИСНОВКИ ТА ПРОПОЗИЦІЇ 58

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 61

ДОДАТКИ 55

					ДТЕУ 121 02-6.МР			
Зм.	*Аркуш*	*№ докум.*	*Підпис*	*Дата*				
Зав. каф.		Криворучко О.В.		01.11.23	*Автоматизована система збору, аналізу та зберігання маркетингових даних з соціальних мереж*	*Стадія*	*Аркуш*	*Аркушів*
Керівник		Жирова Т.О		01.11.23		*Зміст*	*3*	*61*
Гарант		Котенко Н.О.		01.11.23		*Факультет інформаційних технологій 2м курс, 2 група*		
Розробив		Голуб Ю.О.		01.11.23	*Зміст*			

ВСТУП

Актуальність теми полягає в тому що сучасне суспільство характеризується стрімким розвитком інформаційних технологій та широким використанням соціальних мереж у сфері маркетингу. Розуміння та ефективне використання маркетингових даних, які генеруються на платформах соціальних мереж, стає надзвичайно важливим завданням для підприємств та маркетологів. У зв'язку з цим, розробка автоматизованої системи збору, аналізу та зберігання маркетингових даних з соціальних мереж стає актуальною та перспективною темою дослідження в галузі комп'ютерних наук.

Метою даного дослідження є створення потужного та інноваційного інструменту, що використовує мову програмування Python, платформу Google Cloud Platform (GCP), технологію контейнеризації Docker, та базу даних SQL з використанням системи управління базами даних PostgreSQL. Ця система має на меті автоматизувати процес збору, аналізу та зберігання маркетингових даних з різних соціальних мереж.

Об’єктом дослідження є автоматизована система збору, аналізу та зберігання маркетингових даних з соціальних мереж. Дослідження спрямоване на розробку цієї системи, її функціональності та ефективності в контексті збору та аналізу маркетингових даних з соціальних мереж.

Предметом дослідження є використання мови програмування Python, платформи GCP, технології Docker, та бази даних SQL з використанням PostgreSQL, BigQuery для розробки автоматизованої системи збору, аналізу та зберігання маркетингових даних з соціальних мереж. Основними аспектами

дослідження є розробка ефективного алгоритму збору даних з різних платформ соціальних мереж, розробка механізмів для аналізу та визначення ключових показників ефективності маркетингових кампаній, а також створення надійної та масштабованої бази даних для зберігання цих інформаційних ресурсів.

РОЗДІЛ 1

ТЕОРЕТИЧНЕ ДОСЛІДЖЕННЯ МЕХАНІЗМІВ АВТОМАТИЗОВАНОЇ СИСТЕМИ ЗБОРУ, АНАЛІЗУ ТА ЗБЕРІГАННЯ МАРКЕТИНГОВИХ ДАНИХ

1.1. Поняття та теоретичні основи збору маркетингових даних з соціальних мереж

Маркетингові дані в сучасному контексті представляють собою важливу інформацію, що стосується споживачів, ринків та продуктів. Вони включають в себе різноманітні параметри, показники та характеристики, що дозволяють маркетологам краще розуміти потреби та поведінку своєї цільової аудиторії.

Роль маркетингових даних в сучасному маркетингу надзвичайно важлива. Вони є ключовим елементом для прийняття обґрунтованих рішень, розробки ефективних маркетингових стратегій та встановлення зв'язку зі споживачами. Завдяки маркетинговим даним, компанії можуть отримати глибоке розуміння своєї цільової аудиторії, виявити тенденції на ринку, оцінити ефективність своїх продуктів та послуг, а також побудувати персоналізовані маркетингові кампанії.

Одним з ключових аспектів визначення маркетингових даних є їх різноманітність. Вони можуть включати дані про демографічні характеристики споживачів, їхні поведінкові звички, інтереси, покупкові звички, взаємодії в соціальних мережах та багато іншого. Ці дані можуть бути отримані з різних джерел, таких як веб-аналітика, соціальні мережі, опитування, CRM-системи та інші.

Важливою роллю маркетингових даних є їх використання для підтримки прийняття рішень. Аналіз маркетингових даних дозволяє виявляти ключові тренди, залежності та патерни на основі яких можна зробити обґрунтовані висновки і прийняти стратегічні рішення. Маркетингові дані допомагають встановити ефективну комунікацію зі споживачами, адаптувати продукти та послуги до їхніх потреб, а також визначити оптимальні канали просування.

Застосування маркетингових даних також дозволяє проводити цільове сегментування аудиторії, що дозволяє розробляти персоналізовані стратегії маркетингу. Завдяки даним про індивідуальні вподобання та звички споживачів, компанії можуть надавати персоналізовані рекламні пропозиції, пропонувати індивідуальні знижки та спеціальні акції, що сприяє збільшенню лояльності та задоволеності клієнтів.

Важливою характеристикою маркетингових даних є їх об'єм та швидкість збору. Завдяки сучасним інструментам та технологіям, таким як платформа GCP, мова програмування Python та технологія контейнеризації Docker, ми отримуємо можливість автоматизувати процес збору, аналізу та зберігання маркетингових даних. Це дозволяє швидко та ефективно обробляти великі обсяги даних та отримувати актуальну інформацію для прийняття рішень.

У підсумку, маркетингові дані виконують важливу роль у сучасному маркетингу. Вони допомагають розуміти споживачів, аналізувати ринкові тенденції, розробляти персоналізовані стратегії та приймати обґрунтовані рішення. Застосування сучасних технологій, таких як Python.

Розробка системи збору, аналізу та зберігання маркетингових даних з соціальних мереж включає кілька етапів, які гарантують ефективність та надійність системи. Ось декілька основних етапів, які варто враховувати:

1. Визначення вимог: Перший етап полягає у розумінні потреб та вимог до системи збору, аналізу та зберігання маркетингових даних. Це включає

виявлення джерел даних з соціальних мереж, визначення потрібної інформації, збір та збереження даних, а також можливості аналізу й візуалізації.

2. Розробка алгоритму збору даних: Наступним кроком є розробка алгоритму, який дозволить отримати дані з обраного джерела соціальних мереж. За допомогою мови програмування Python та відповідних бібліотек, таких як Requests, BeautifulSoup або Selenium, можна здійснити запити до API соціальних мереж, або використовувати веб-скрапінг для отримання потрібної інформації з веб-сторінок.
3. Обробка та аналіз даних: Після збору даних необхідно провести їх обробку та аналіз. За допомогою бібліотеки Pandas можна виконати різноманітні операції над даними, такі як фільтрація, сортування, групування та обчислення статистичних показників. Також можна використовувати бібліотеки для візуалізації даних, такі як Matplotlib або Seaborn, для створення графіків та діаграм, що допоможуть зрозуміти тенденції та залежності в даних.
4. Зберігання даних: Важливим етапом є вибір та реалізація механізму зберігання даних. В даному випадку, ви можете використати систему керування базами даних (СКБД) для зберігання маркетингових даних. PostgreSQL є одним з популярних СКБД, який надає широкий функціонал та надійність. Використання PostgreSQL дозволить створити структуровані таблиці для зберігання даних та виконувати запити для отримання необхідної інформації.
5. Автоматизація процесу: Щоб забезпечити ефективну та стабільну роботу системи, важливо автоматизувати процеси збору, аналізу та зберігання даних. Для цього можна використати мову програмування Python та інструменти, такі як планувальник завдань або контейнеризація за допомогою Docker. Завдяки автоматизації, ви

зможете запускати процеси збору даних за розкладом, автоматично аналізувати нові дані та зберігати їх у базу даних без необхідності ручного втручання.

6. Забезпечення безпеки: У контексті збору та зберігання маркетингових даних з соціальних мереж, безпека є важливою складовою. Варто забезпечити захист інформації, використовуючи різноманітні методи, такі як шифрування даних, контроль доступу та захист від несанкціонованого використання. Також важливо дотримуватись політик безпеки соціальних мереж та використовувати API та інструменти згідно з їхніми правилами.

Обрання середовища розробки.

Visual Studio Code – це повнофункціональний текстовий редактор для редагування локальних файлів чи бази коду. Він включає різні функції для редагування бази коду, яка допомагає розробникам відстежувати зміни. Різні функції, що підтримуються в VS Code:

1. Підсвічування синтаксису.
2. Авто відступом.
3. Розпізнавання типів файлів.
4. Бічна панель із файлами вказаного каталогу.
5. Макрос.
6. Плагін та пакети.
7. Vs Code використовується як інтегрований редактор розробки (IDE), як Sublime і NetBeans. Поточна версія редактора Vs Code сумісна з різними операційними системами, такими як Windows, Linux та MacOS.

Обрання мови програмування.

Python — ця мова програмування є однією з найпопулярніших мов у сфері програмування, особливо у контексті наукових досліджень та аналізу даних. Існує кілька ключових причин, чому ми обрали Python для реалізації нашої системи збору, аналізу та зберігання маркетингових даних з соціальних мереж.

Головні переваги мови програмування Python:

- Python — це легка інтерпретована мова програмування;
- Python є дуже простою та зрозумілою мовою програмування;
- Python має велику кількість розширень та бібліотек;
- Python має велику кількість розширень та бібліотек, які допомагають в роботі;
- Python є платформонезалежною мовою, що означає, що ви можете розробляти програми на Python для різних операційних систем.

Крім того, Python має велику активну спільноту розробників, що забезпечує доступ до багато ресурсів, форумів та документації. Це дає можливість швидко знайти відповіді на питання та розв'язати проблеми під час розробки системи.

Обрання бази даних.

PostgreSQL є однією з найпопулярніших та потужних систем управління базами даних, яка знаходить широке застосування в сфері розробки програмного забезпечення та аналізу даних, включаючи нашу тему - систему збору, аналізу та зберігання маркетингових даних з соціальних мереж.

PostgreSQL пропонує багато переваг, які роблять його привабливим в контексті нашого дослідження. По-перше, він є відкритим джерелом, що означає, що ви маєте доступ до його вихідного коду та можете вносити зміни

в систему відповідно до своїх потреб. Це забезпечує гнучкість та можливість налаштовувати базу даних під конкретні вимоги проекту.

По-друге, PostgreSQL є надійною та стабільною системою, яка володіє високим рівнем цілісності та надійності даних. Вона підтримує транзакційну безпеку, механізми реплікації та відновлення, що дозволяє забезпечити доступність та захищеність даних.

По-третє, PostgreSQL має потужний мовний ряд SQL, що робить його ідеальним вибором для роботи зі структурованими даними, які ми збираємо з соціальних мереж. Він підтримує розширення SQL стандарту, а також має розширення для роботи з JSON, географічними даними та іншими типами даних, які можуть бути корисними в контексті маркетингових даних.

BigQuery - це високоефективна, повністю керована база даних від Google Cloud, спеціально розроблена для аналізу великих обсягів даних. Вона вирізняється надзвичайною масштабованістю, дозволяючи легко обробляти і аналізувати терабайти і петабайти даних без необхідності в складних конфігураціях або апаратному обслуговуванні. Швидкість запитів у BigQuery вражає, здатність виконувати складні аналітичні завдання всього за декілька секунд дозволяє зосередитися на важливих висновках з даних. Використання стандартного SQL робить запити і аналітику зрозумілими для широкого спектра користувачів. Однією з ключових переваг є також відсутність необхідності в адмініструванні - BigQuery автоматично керує налаштуваннями і масштабуванням. Також, завдяки вбудованим інструментам аналізу, підтримці машинного навчання та інтеграції з іншими сервісами Google Cloud, можна досліджувати дані глибше та впроваджувати нові можливості для роботи.

Головною перевегою BigQuery є масштабованість і вона ідеально пдходить для нашої задачі в доповнення з PostgreSQL.

Контейнеризація.

Docker є потужним інструментом контейнеризації, який відіграє важливу роль у нашому дослідженні системи збору, аналізу та зберігання маркетингових даних з соціальних мереж. Використання Docker дозволяє нам пакувати нашу програмну систему та всі її залежності у контейнери, що є самодостатніми ізольованими середовищами. Це забезпечує консистентність і переносимість нашого додатку незалежно від середовища, на якому він працює.

Завдяки Docker ми можемо швидко та легко розгортати нашу систему на різних серверах або хмарних платформах, забезпечуючи її масштабованість та доступність. Контейнеризація дозволяє нам уникнути проблем, пов'язаних зі сумісністю програмного забезпечення та конфліктами між залежностями. Ми можемо легко керувати та масштабувати нашу систему, додаючи або видаляючи контейнери за необхідності.

Docker також забезпечує нам можливість ізольовано працювати з базою даних, використовуючи контейнери, що мають власне середовище для PostgreSQL. Це сприяє забезпеченню безпеки та незалежності наших даних, а також спрощує процес розгортання та управління базою даних.

Узагальнюючи, використання Docker у нашому дослідженні дозволяє нам забезпечити легку, ефективну та масштабовану розгортку нашої системи збору, аналізу та зберігання маркетингових даних, а також забезпечити ізольованість та стабільність середовища для бази даних PostgreSQL.

Google Cloud Compute Engine - це інфраструктурна послуга у хмарі, яка надає можливість користувачам створювати та управляти віртуальними машинами (VM) на інфраструктурі Google. Ця послуга дозволяє розгортати віртуальні сервери з певною кількістю обчислювальних ресурсів, пам'яті, обсягу диску та інших параметрів згідно з власними потребами.

Основні характеристики та можливості Google Cloud Compute Engine:

- **Віртуальні машини**: Ми можемо створювати віртуальні машини на базі різних операційних систем, таких як Linux та Windows. Вам доступний великий вибір типів VM з різними розмірами ресурсів для відповідності вашим вимогам.
- **Скейлінг**: Можна змінювати розмір та кількість віртуальних машин в залежності від навантаження. Це дозволяє ефективно використовувати ресурси із забезпеченням високої доступності та продуктивності.
- **Мережеві можливості**: Compute Engine надає можливість налаштовувати мережі, правила файрволів, зовнішні та внутрішні IP-адреси для ваших віртуальних машин.
- **Додаткові служби**: Можна використовувати додаткові послуги Google Cloud, такі як моніторинг, управління ідентифікацією та доступом, збереження даних тощо.
- **Продуктивність**: Google використовує власні технології віртуалізації та оптимізації для забезпечення високої продуктивності та швидкодії ваших віртуальних машин.
- **Безпека**: Compute Engine надає різноманітні можливості для захисту ваших данних та віртуальних машин, включаючи шифрування, файрволи та інші заходи безпеки.
- **Шаблони і зразки**: Ми можемо створювати шаблони віртуальних машин, щоб швидко розгортати однотипні сервери або навіть використовувати готові зразки від Google.
- **Автоматизація**: Можна автоматизувати управління віртуальними машинами через API, інструменти командного рядка та консолі керування Google Cloud.

- **Ціноутворення**: Compute Engine має гнучку систему ціноутворення, де ви платите лише за використані ресурси за години, безпосередньо контролюючи свої витрати.

Google Cloud Compute Engine дозволяє підприємствам та розробникам масштабувати та управляти обчислювальними ресурсами у хмарі для різних завдань, від веб-хостингу та розгортання додатків до обробки великих обсягів даних та наукових досліджень.

Google Cloud Functions - це обчислювальна послуга у хмарі, яка дозволяє створювати та виконувати код у відповідь на події в хмарному середовищі, без необхідності вручну керувати інфраструктурою. Ця послуга базується на концепції "функцій", які є невеликими фрагментами коду, що виконуються при виникненні певних подій або викликів API.

Основні характеристики та можливості Google Cloud Functions:

- **Події та спускові механізми**: Можна налаштовувати функції для виконання при виникненні різних подій, таких як завершення завдань у хмарному сховищі, сповіщення від служби Google Assistant, отримання повідомлень в черзі подій тощо.
- **Підтримка мов програмування**: Google Cloud Functions підтримує кілька популярних мов програмування, таких як Node.js, Python, Go, Java та .NET. Це дає вам можливість використовувати мову, з якою ви найкраще знайомі.
- **Безстічний масштабування**: Функції автоматично масштабуються згідно з навантаженням. Це означає, що ви можете впевнено відповідати на будь-яке навантаження без необхідності подумати про інфраструктуру.

- **Автоматизована управління інфраструктурою**: Ми не потребуємо вручну налаштовувати віртуальні машини або іншу інфраструктуру. Все це робиться автоматично під час запуску функцій.
- **Швидкість виконання**: Функції запускаються дуже швидко, оскільки вони компілюються та готуються до виконання заздалегідь.
- **Інтеграція з іншими сервісами Google Cloud**: Можена легко поєднувати функції з іншими сервісами Google Cloud, такими як зберігання, бази даних, аналітика тощо.
- **Локальне тестування**: Ми можете тестувати функції локально перед їх розгортанням у хмарі, що допомагає відлагоджувати код.
- **Автентифікація та авторизація**: Ми можете налаштовувати управління доступом до ваших функцій, забезпечуючи безпеку ваших даних.

Google Cloud Functions може бути використаний для реалізації різноманітних сценаріїв, таких як автоматична обробка завдань, відповіді на виклики API, створення мікросервісів та багато іншого. Використання функцій дозволяє фокусуватися на написанні коду та бізнес-логіці, мінімізуючи витрати часу на керування інфраструктурою.

1.2. Аналіз відомих методів систем збору, аналізу та зберігання даних

Під час аналізу ми зосереджуємося на розгляді різних аспектів систем збору, аналізу та зберігання даних, зокрема їх функціональності, продуктивності, масштабованості, безпеки та зручності використання. Ми досліджуємо різні методи та підходи, які використовуються для обробки та аналізу великих обсягів даних, включаючи методи машинного навчання, статистичний аналіз, глибинне навчання та інші.

Аналіз відомих методів допомагає нам визначити оптимальний набір технологій, методів та інструментів, які задовольняють вимоги нашої системи

Він допомагає нам зрозуміти, які підходи та рішення можуть бути ефективними для збору, аналізу та зберігання маркетингових даних з соціальних мереж. Базуючись на результаті аналізу, ми можемо визначити найкращу архітектуру та інструменти для розробки нашої архітектури.

У контексті нашої теми, є кілька методів збору даних з соціальних мереж, які можуть бути використані для отримання маркетингових даних.

API соціальних мереж. Багато популярних соціальних мереж, таких як Facebook, Twitter, Telegram, Instagram та LinkedIn, надають API (інтерфейс програмування додатків), які дозволяють розробникам отримувати доступ до даних цих платформ. Використання API дозволяє автоматично отримувати даних, таких як повідомлення, профілі користувачів, лайки, коментарі та інші, з соціальних мереж для подальшого аналізу.

API (Application Programming Interface) соціальних мереж є набором інструментів, протоколів та правил, які надають розробникам доступ до функціональності та даних соціальних мереж. Використання API дозволяє здійснювати автоматичне взаємодію з платформою соціальної мережі, отримувати дані, надсилати запити та виконувати дії в рамках обмежень, встановлених соціальною мережею.

У контексті нашої теми, використання API соціальних мереж є важливим для збору маркетингових даних з соціальних мереж. За допомогою API ми можемо отримати доступ до різних типів даних, таких як публічні повідомлення, інформацію про користувачів, коментарі, лайки, демографічні дані та інші параметри, які можуть бути корисними для аналізу маркетингової стратегії.

API соціальних мереж зазвичай вимагають аутентифікації, щоб перевірити, що запити до API робляться від імені валідного користувача або додатку. Це може включати використання ключів API або протоколів авторизації, таких як OAuth. Після аутентифікації ми можемо використовувати

API для отримання даних у реальному часі або виконання спеціалізованих запитів.

API соціальних мереж також можуть надавати можливості аналізу даних, такі як статистика взаємодії з повідомленнями, аналітика аудиторії, виявлення трендів та інше. Ці можливості дозволяють нам отримувати інсайти та використовувати дані для покращення маркетингових стратегій і розуміння поведінки наших цільових аудиторій в соціальних мережах.

Веб-скрапінг є методом отримання даних з веб-сторінок шляхом аналізу HTML-коду. У контексті нашої теми, веб-скрапінг може бути використаний для збору маркетингових даних зі сторінок соціальних мереж.

Процес веб-скрапінгу починається зі завантаження веб-сторінки, зазвичай за допомогою HTTP-запиту. Потім, отриманий HTML-код сторінки проходить обробку і аналіз для витягування необхідних даних. Для аналізу HTML-коду зазвичай використовуються спеціальні бібліотеки або інструменти, такі як BeautifulSoup у Python. За допомогою веб-скрапінгу можна отримати різноманітну інформацію зі сторінок соціальних мереж, включаючи текстові повідомлення, дані профілів користувачів, коментарі, лайки, дати публікацій та інші параметри, які можуть бути цінними для аналізу маркетингових даних.

Варто відзначити, що деякі соціальні мережі можуть мати обмеження або блокувати веб-скрапінг через свої політики. Тому важливо перевірити та дотримуватись правил і обмежень, встановлених соціальною мережею, перед використанням веб-скрапінгу.

Також важливо враховувати, що структура HTML-коду сторінок соціальних мереж може змінюватись з часом, тому потрібно регулярно оновлювати аналізатори інформації і перевіряти, чи продовжує веб-скрапінг працювати на актуальних сторінках.

Спільноти дослідників (скоріш всього не підійде в нашому випадку). У деяких випадках, дослідники можуть запрошувати користувачів соціальних мереж до участі в дослідженнях або анкетах. Це може включати запити на заповнення анкет, участь у фокус-групах або надання доступу до своїх профілів для дослідження. Цей метод дозволяє отримати відповіді користувачів безпосередньо від них і зібрати маркетингові дані. Ці спільноти об'єднують фахівців, дослідників та експертів з різних сфер, які мають спільний інтерес до вивчення та розвитку цієї області. У спільнотах дослідників зазвичай відбувається обмін ідеями, результатами досліджень, новими методиками та підходами до збору, аналізу та зберігання маркетингових даних з соціальних мереж. Члени спільноти можуть презентувати свої дослідження, виступати з доповідями, ділитись кейсами використання та передовими практиками.

Такі спільноти можуть існувати як в онлайн-середовищі, де фахівці обговорюють тематичні питання на форумах, вебінарах, блогах або спільних проектах, так і офлайн, де проводяться конференції, семінари, наукові зустрічі та студійні групи.

1.2. Характеристика існуючих методів та програмних засобів збору аналізу та зберігання маркетингових даних

Збір даних методом **використання API** (інтерфейсу програмування додатків) є широко використовуваним та ефективним способом отримання маркетингових даних з соціальних мереж. API надає програмістам можливість взаємодіяти з веб-сервісами та отримувати доступ до даних, які надаються власниками платформи.

Використання API для збору маркетингових даних з соціальних мереж є надзвичайно корисним та ефективним підходом. Цей метод дозволяє отримати доступ до широкого спектру функціональності, такої як отримання

профілів користувачів, публікацій, коментарів тощо, що допомагає збирати різноманітні дані для проведення аналізу та розробки маркетингових стратегій. API надають структуровані дані у зручному форматі, що спрощує їх подальшу обробку та аналіз. Крім того, використання API забезпечує безпеку та захист даних завдяки механізмам автентифікації, а також гарантує надійність та стабільність доступу до даних. Власники платформи надають документацію та ключі доступу, що спрощує процес використання API та підвищує його доступність для розробників.

Збір даних методом **веб-скрапінгу** полягає у витягуванні інформації з веб-сторінок шляхом програмного аналізу HTML-коду. Цей метод використовується для отримання даних, які не надаються через публічні API або не можуть бути легко доступні через інші засоби.

Основні характеристики збору даних методом веб-скрапінгу:

- Аналіз HTML-коду: Веб-скрапінг передбачає аналіз HTML-коду веб-сторінок, що вимагає розуміння структури та розмітки сторінки. Зазвичай використовуються спеціальні бібліотеки, такі як BeautifulSoup або Scrapy, для зручного розбору HTML та витягування потрібних даних.
- Вибір селекторів: Для витягування конкретних даних з HTML-коду необхідно визначити правильні селектори. Селектори це шляхи до елементів HTML, такі як теги, класи, ідентифікатори тощо. Використовуючи правильні селектори, ви можете точно вказати, які елементи ви хочете витягнути.
- Обробка даних: Після витягування даних з веб-сторінки вони зазвичай потребують подальшої обробки. Це може включати видалення зайвих символів, очищення від HTML-тегів або перетворення у зручний формат, наприклад, у JSON або CSV.
- Обмеження доступу та правові аспекти: При веб-скрапінгу важливо враховувати правові аспекти та обмеження доступу до даних. Деякі

веб-сайти можуть мати політику, яка забороняє скрапінг або обмежує кількість запитів, які можна виконати.

Веб-скрапінг є потужним інструментом для збору даних з веб-сторінок, який надає доступ до недоступних інформаційних ресурсів і дозволяє отримувати оновлені дані в реальному часі. Цей метод дозволяє отримувати інформацію, яка не надається через публічні API або інші доступні засоби, і забезпечує гнучкість і адаптабельність у витягуванні потрібних даних. Проте, використання веб-скрапінгу має свої обмеження та виклики. Перш за все, веб-скрапінг може порушувати правила власників веб-сайтів та бути суперечливим з юридичною точки зору. Деякі веб-сайти можуть накладати обмеження на кількість запитів або використання скрапінгу взагалі.

Крім того, веб-скрапінг вимагає розуміння структури і розмітки HTML-коду та використання правильних селекторів для витягування потрібних даних. Цей процес може бути складним і часомістким, особливо при змінах у веб-сторінці. Також варто враховувати, що веб-скрапінг не є універсальним методом збору даних. Деякі веб-сайти можуть застосовувати захисні механізми, що ускладнюють або унеможливлюють скрапінг. В таких випадках може знадобитися альтернативний підхід або використання публічних API, якщо доступні.

Для зберігання даних про публікації користувачів, може бути створена таблиця "Публікації", де ми зберігаємо інформацію про текст публікації, дату публікації, ідентифікатор користувача, до якого вона відноситься та інші відповідні дані.

Таким чином, за допомогою таблиць і відповідних стовпців у базі даних, ми організовуємо та зберігаємо отримані маркетингові дані. Кожен запис в таблиці відповідає конкретному об'єкту, такому як користувач або публікація, і містить відповідні атрибути та значення.

Запити до бази даних здійснюються за допомогою SQL, де можна виконувати операції вибору, оновлення, видалення та вставки даних. Це дозволяє нам легко взаємодіяти з базою даних і отримувати потрібну інформацію для аналізу та обробки маркетингових даних.

Збереження даних в базі даних дозволяє забезпечити надійність, безпеку та швидкий доступ до маркетингових даних, що є критичними для успішної роботи автоматизованої системи збору, аналізу та зберігання маркетингових даних. Бази даних, такі як PostgreSQL, також надають можливості резервного копіювання та відновлення даних, що забезпечує їх безпеку та захист від втрати.

Збереження маркетингових даних в базі даних також дозволяє здійснювати складні запити та аналізувати дані для виявлення трендів, патернів та залежностей. Можна використовувати мову запитів SQL для фільтрації, сортування, агрегації даних та створення звітів і графіків, що допомагає приймати обґрунтовані рішення щодо маркетингових стратегій.

Окрім того, зберігання даних в базі даних спрощує спільну роботу та обмін даними між різними членами команди. Бази даних можуть бути розгорнуті на серверах і доступні з різних пристроїв, що дозволяє команді працювати з однією актуальною версією даних та забезпечує їх синхронізацію.

Загалом, зберігання маркетингових даних в базі даних, зокрема за допомогою SQL і PostgreSQL, забезпечує структуроване, надійне та безпечне зберігання даних, швидкий доступ до інформації та можливості для аналізу та прийняття обґрунтованих маркетингових рішень. Це сприяє ефективній роботі автоматизованої системи збору, аналізу та зберігання маркетингових даних з соціальних мереж.

1.4. Висновки до розділу 1

Роль маркетингових даних у сучасному маркетингу надзвичайно важлива. Концепції та теоретичні основи збору маркетингових даних із соціальних мереж. API соціальних мереж. Вони можуть включати дані про демографічні показники споживачів, поведінкові звички, інтереси, купівельні звички, взаємодію в соціальних мережах тощо. Одним із ключових аспектів визначення маркетингових даних є їх різноманітність. Використання маркетингових даних також дозволяє здійснювати цільову сегментацію аудиторії, що дозволяє розробляти персоналізовані маркетингові стратегії.

Маркетингові дані в сучасному контексті представляють важливу інформацію про споживачів, ринки та продукти. Вони є ключовим елементом для прийняття обґрунтованих рішень, розробки ефективних маркетингових стратегій і встановлення контактів із споживачами.

Зберігання даних: Важливим етапом є вибір і впровадження механізму зберігання даних роль маркетингових даних полягає в тому, щоб використовувати їх для підтримки прийняття рішень. Аналіз маркетингових даних дозволяє виявити ключові тенденції, залежності та закономірності, на основі яких можна робити обґрунтовані висновки та приймати стратегічні рішення. Ці дані можна отримати з різних джерел, таких як веб-аналітика, соціальні мережі, опитування, системи CRM та інші. Завдяки сучасним інструментам і технологіям, таким як платформа GCP, мова програмування Python і технологія контейнеризації Docker, ми отримуємо можливість автоматизувати процес збору, аналізу та зберігання маркетингових даних.

Підсумовуючи, маркетингові дані відіграють важливу роль у сучасному маркетингу. Вони включають різні параметри, показники та характеристики, які дозволяють маркетологам краще зрозуміти потреби та поведінку своєї цільової аудиторії. Маркетингові дані допомагають налагодити ефективну комунікацію зі споживачами, адаптувати продукти та

послуги до їхніх потреб, визначити оптимальні канали просування. Важливою характеристикою маркетингових даних є їх обсяг і швидкість збору. Завдяки маркетинговим даним компанії можуть отримати глибоке розуміння своєї цільової аудиторії, визначити ринкові тенденції, оцінити ефективність своїх продуктів і послуг і створити персоналізовані маркетингові кампанії.

РОЗДІЛ 2

ПРОЕКТУВАННЯ СИСТЕМИ ЗБОРУ, АНАЛІЗУ ТА ЗБЕРІГАННЯ МАРКЕТИНГОВИХ ДАНИХ

2.1. Постановка задачі

Постановка задачі передбачає ретельне визначення функціональних вимог системи, яка повинна забезпечувати зручну інтеракцію з API соціальних мереж, автоматичний збір та оновлення даних, аналіз метрик та тенденцій, збереження і безпеку маркетингових даних, а також можливість генерації звітів та візуалізацію результатів.

Важливо також враховувати використання сучасних технологій, таких як Python для програмування, Google Cloud Platform (GCP) для розгортання системи, Docker для контейнеризації та PostgreSQL для зберігання даних. Це дозволить створити масштабовану, гнучку та ефективну систему, яка відповідатиме потребам бізнесу та забезпечуватиме високу якість обробки та аналізу маркетингових даних.

Для розробки автоматизованої системи збору, аналізу та зберігання маркетингових даних з використанням

1.Збір даних з соціальних мереж:

- Розробити модуль збору даних, який буде взаємодіяти з API або веб-скрапінгом різних соціальних мереж (наприклад, Facebook, Twitter, Instagram і т.д.).
- Забезпечити можливість аутентифікації та отримання доступу до даних користувачів через API.

- Запланувати автоматичні процеси збору даних за певний часовий період або на підставі заданих подій, наприклад, нових публікацій або коментарів.

2. Аналіз маркетингових даних:

- Розробити алгоритми аналізу даних для виявлення ключових метрик та інформації зі зібраних даних.
- Забезпечити можливість фільтрації, сортування та групування даних для подальшого детального аналізу.
- Реалізувати функції виявлення тенденцій, співставлення даних з попередніми періодами та прогнозування результатів маркетингових кампаній.

3. Зберігання маркетингових даних:

- Створити базу даних в PostgreSQL для зберігання маркетингових даних.
- Розробити механізми зберігання та оновлення даних у базі даних з використанням SQL-запитів.
- Забезпечити можливість резервного копіювання та відновлення даних для запобігання втрати інформації

4. Інтеграція з GCP та Docker:

- Використовувати Google Cloud Platform (GCP) для розгортання та керування системою.
- Створити контейнеризовану версію системи з використанням Docker для забезпечення продуктивності та легкості розгортання.

5. Візуалізація та звітність:

- Розробити інтерфейс для візуалізації маркетингових даних, що дозволяє створювати графіки, діаграми та звіти на основі аналізованих даних.
- Забезпечити можливість експорту даних та звітів у різних форматах (наприклад, PDF, CSV) для подальшого використання або розповсюдження.

6.Забезпечення безпеки даних:

- Реалізувати механізми захисту маркетингових даних від несанкціонованого доступу, шифрування та контролю доступу до системи.
- Забезпечити резервне копіювання та відновлення даних для запобігання втрати інформації.

2.2. Метод ETL або ELT пайплайнів

Головним завданням **ETL** та **ELT**-систем є структурування, збагачення, оптимізація та передача вихідних даних компанії з кількох програмних оболонок у єдину централізовану базу зберігання для подальшої обробки.

Спочатку визначимо, **що таке ETL**. Це модель, яка працює за принципом Extract, Transform, Load. Інформація, що отримується з різних джерел, оброблена відповідно до алгоритмів довідників різних IT-систем і має неоднаковий ступінь деталізації, наводиться в єдиний формат і стає придатною для подальшої обробки. Наступне завдання **ETL** – це доставка даних, яка має здійснюватися найшвидшим способом без втрати якості та достовірності.

ELT-системи відрізняються послідовністю виконуваних дій: Extract, Load, Transform. Дані спочатку отримують та завантажують, а лише після обробляють. Модель **ELT** працює з величезними масивами асинхронних даних, розвиваючи вражаючу швидкість. Вона використовує ресурси хмарних обчислень, які можна масштабувати без використання локального обладнання. Таким чином, **ELT** має на увазі не тільки зміну послідовності дій, але також якісно новий підхід до процесу трансформації даних.

Як працює ETL:

ETL здійснює підготовку даних для подальшої аналітики одразу після її вивантаження. Для цього система отримує інформацію з різних баз, очищає від помилок, призводить до єдиного формату та рівня деталізації. Таким чином, збагачені дані, що одержуються з різних джерел, наводяться до єдиного формату, що дозволяє ефективно з ними взаємодіяти. Тільки після всіх цих маніпуляцій інформація надходить до цільової репозиторії і стає доступною для вивчення з використанням технологій BI та data science.

Рис. 2.1. Принцип роботи ETL пайплайну

Головними перевагами **ETL** є:

- добре вивчений процес, який легко підтримувати на професійному рівні. Існує велика кількість перевірених часом інструментів та платформ **ETL**;
- якісна підготовка даних для аналізу – після вивантаження інформації вона може бути використана для вивчення без додаткового коригування;
- наявність аудиторського сліду, що дозволяє відстежувати походження даних.

ETL якісно впорядковує вихідну інформацію, готуючи її до процесів візуалізації, моделювання та подальшого перетворення.

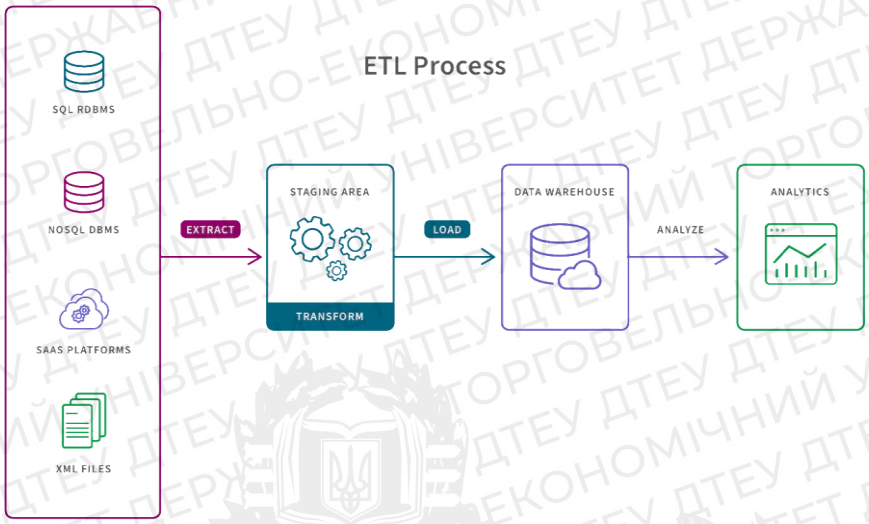

Рис. 2.2. Модель роботи ETL пайплайну

Як працює ELT:

В епоху постійного збільшення кількості джерел інформації та обсягу даних, які можуть бути використані для подальшого аналізу, бізнес потребує прискорення процесів їх підготовки та передачі. Швидкість стає важливою конкурентною перевагою, тому впровадження систем **ELT** є все більш актуальним питанням.

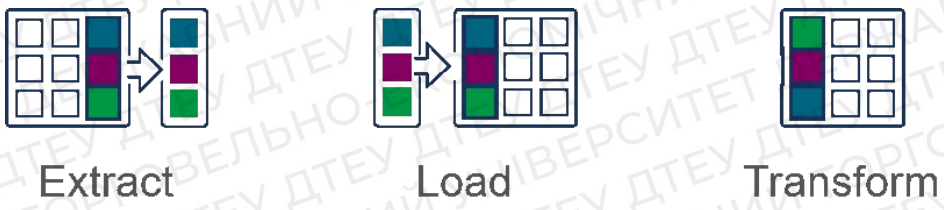

Рис. 2.3. Принцип роботи ELT пайплайну

Прискорення процесу передачі досягається з допомогою оптимізації підходу внаслідок зміни послідовності операцій. Спочатку дані виймаються та завантажуються, а лише після обробляються. У процесі їх трансформації

задіяні хмарні технології, що дозволяє оптимізувати швидко і непомітно для користувача.

Особливістю **ELT** є також поступове опрацювання інформації, що здійснюється в міру формування запитів. При цьому користувач може використовувати єдиний репозиторій для різних програм. На відміну від процесу підготовки інформації в конвеєрі, вихідні дані не будуть втрачені – для вирішення конкретного завдання створюється репліка. Наявність подібної проміжної бази даних дозволяє повертатися до вивчення вихідної інформації неодноразово, не вносячи технічних змін до **ELT**-системи.

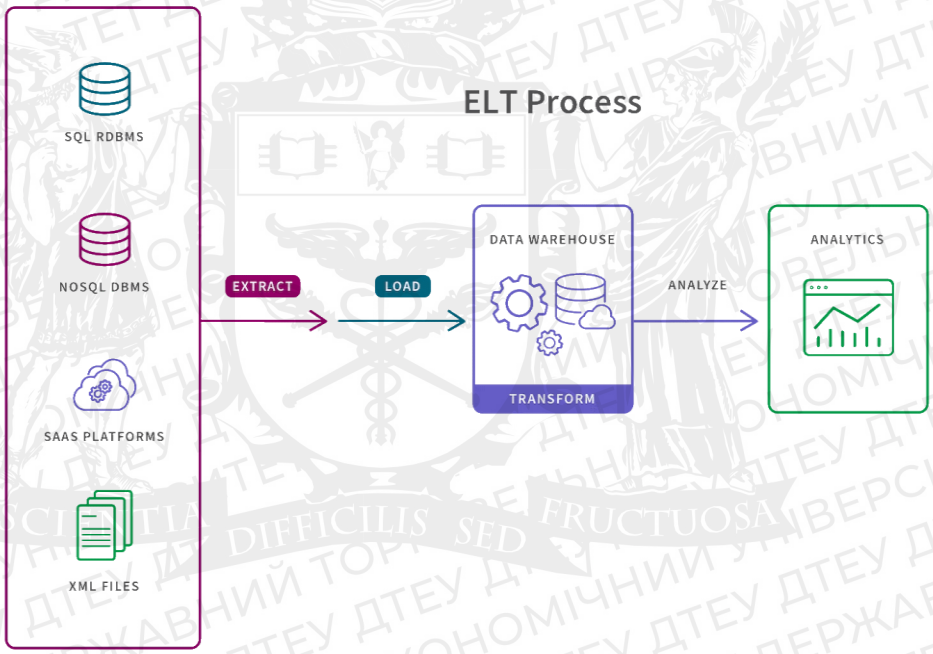

Рис. 2.4. Модель роботи ELT пайплайну

Завдяки використанню ELT ви зможете оперативно керувати великими або постійно зростаючими обсягами даних. До головних переваг такої системи відносять наступні пункти:

- Висока швидкість роботи. По-перше, процесу передачі не передує їх попередня підготовка всередині системи. По-друге, з інформацією працюють хмарні послуги, призначені для швидкої взаємодії з великими

обсягами даних. Інтерактивна аналітика самообслуговування стає доступною як реального часу.

- Гнучкість. Дані перетворюються відповідно до запитів конкретних користувачів у момент безпосереднього звернення та можуть використовуватися для різних цілей багаторазово. При цьому не доведеться настроювати процеси на рівні ELT-системи.
- Масштабованість. Збільшення обсягів даних не є проблемою для ELT або хмарного сховища.
- Прозорість. Користувачі оперативно одержують інформацію про те, які дані доступні для вивчення.
- Низькі експлуатаційні витрати. При використанні хмарного ELT немає необхідності інвестувати в локальне обладнання та доопрацьовувати систему у разі розширення потреб бізнесу в аналітиці або збільшення обсягів даних. Ви платите тільки за обсяг хмарних сервісів, що використовується.

Завдяки впровадженню ELT можна скоротити витрати фінансів та часу, необхідних для обробки інформації, а також отримуєти більше можливостей для використання вихідних даних.

2.3. Метод MPP (Massively Parallel Processing)

Масивна паралельна обробка (MPP) - це сучасний архітектурний підхід у сфері обробки даних, спрямований на розв'язання завдань обчислення та аналізу великого обсягу даних. Основна ідея MPP полягає в тому, щоб розділити складні завдання на більш прості підзавдання та виконувати їх одночасно на багатьох обчислювальних вузлах. Це дає можливість ефективно використовувати паралельні обчислення та розподілити навантаження між різними ресурсами.

У контексті Google BigQuery як хмарному сервісу аналізу даних, який використовує архітектуру масивної паралельної обробки (MPP) для ефективної обробки великих обсягів даних. Ця архітектура дозволяє розподілити дані на менші частини та обробляти їх одночасно на різних обчислювальних вузлах. Запити SQL розбиваються на дрібніші підзапити, які виконуються паралельно на кількох вузлах. Результати цих підзапитів потім об'єднуються, надаючи кінцевий результат запиту. Важливою рисою BigQuery є його здатність автоматично масштабувати обчислювальні ресурси відповідно до розміру даних та складності запитів. Такий підхід допомагає досягти високої швидкості та продуктивності обробки даних, що робить BigQuery ефективним інструментом для аналізу великих обсягів даних.

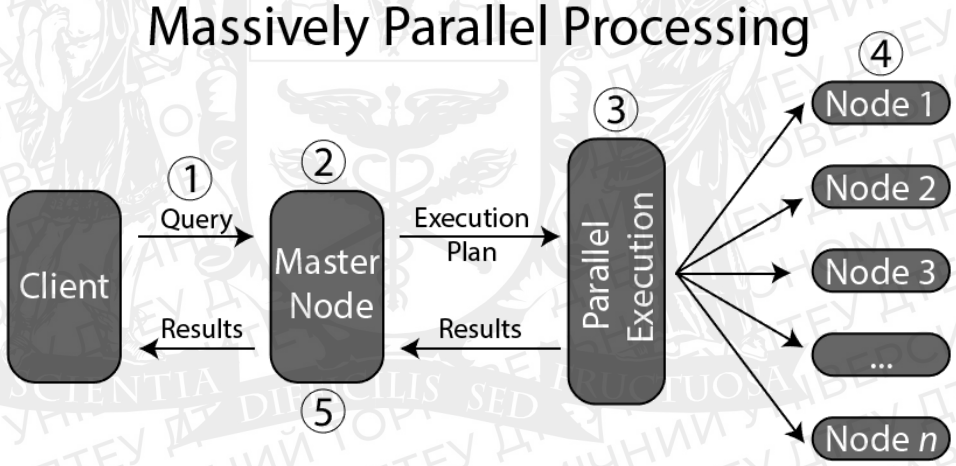

Рис. 2.5. Модель роботи “Масивної паралельної обробки”

В MPP-архітектурі дані розбиваються на частини, які називаються "частковими даними". Кожний обчислювальний вузол отримує свій набір часткових даних для обробки. Після завершення обробки кожен вузол об'єднує результати, і в результаті отримуємо завершений відповідь. Цей процес дозволяє ефективно розподіляти завдання між вузлами та прискорювати обчислення.

Однією з головних переваг MPP є горизонтальне масштабування. Це означає, що для збільшення продуктивності та потужності можна додавати

нові обчислювальні вузли, що відносно легко реалізовується. Це особливо важливо для великих обсягів даних, де попит на обчислювальні ресурси може різко змінюватися.

MPP також забезпечує високу доступність, оскільки вузли можуть працювати незалежно. Якщо один вузол виходить з ладу, інші можуть продовжувати обробку. Це підвищує надійність системи та забезпечує безперервність обчислень.

У підсумку, MPP дозволяє оптимально використовувати паралельні обчислення та ефективно обробляти великі обсяги даних шляхом розподілу завдань між обчислювальними вузлами. Ця архітектура використовується для різноманітних завдань, включаючи аналіз даних, бізнес-інтелект, наукові дослідження та інші області, де швидкість та ефективність обробки є ключовими факторами.

2.4. OLTP та OLAP методи та системи БД

Як працює OLTP:

OLTP (online transactional processing) забезпечує швидку і точну обробку даних, що лежать в основі банкоматів та онлайн-банкінгу, касових апаратів та електронної комерції, а також десятків інших сервісів, з якими ми взаємодіємо щодня. OLTP, або обробка транзакцій в режимі реального часу, дозволяє виконувати велику кількість транзакцій з базами даних в режимі реального часу великою кількістю людей, як правило, через Інтернет.

Транзакція бази даних - це зміна, вставка, видалення або запит даних у базі даних. OLTP-системи (і транзакції з базами даних, які вони підтримують) є рушійною силою багатьох фінансових операцій, які ми здійснюємо щодня, включаючи банківські операції в Інтернеті та через банкомати, електронну комерцію та покупки в магазинах, бронювання готелів та авіаквитків, і це лише деякі з них. У кожному з цих випадків транзакція в базі даних також

залишається як запис про відповідну фінансову операцію. OLTP також може керувати обміном нефінансовими базами даних, включаючи зміну паролів і текстові повідомлення.

В OLTP загальною, визначальною характеристикою будь-якої транзакції бази даних є її атомарність (або неподільність) - транзакція або завершується успішно, або зазнає невдачі (або скасовується). Вона не може залишатися в очікуванні або проміжному стані.

Загалом, OLTP-системи роблять наступне:

- Обробляють велику кількість відносно простих транзакцій: Зазвичай це вставки, оновлення та видалення даних, а також прості запити до даних (наприклад, перевірка балансу в банкоматі).
- Надавати доступ до одних і тих самих даних багатьом користувачам, забезпечуючи при цьому цілісність даних: OLTP-системи покладаються на алгоритми паралельності, щоб гарантувати, що два користувачі не можуть одночасно змінювати одні й ті ж дані, і що всі транзакції виконуються в правильному порядку. Це запобігає подвійному бронюванню одного й того ж номера в системах онлайн-бронювання та захищає власників спільних банківських рахунків від випадкових овердрафтів.
- Основне це дуже швидка обробка, час відгуку вимірюється мілісекундами: Ефективність OLTP-системи вимірюється загальною кількістю транзакцій, які можна виконати за секунду.
- Надавати індексовані набори даних: Використовуються для швидкого пошуку, вилучення та запитів.
- Доступні 24/7/365: Знову ж таки, OLTP-системи обробляють величезну кількість одночасних транзакцій, тому будь-яка втрата даних або простої можуть мати значні і дорогі наслідки. Повна резервна копія даних повинна бути доступна в будь-який момент часу. OLTP-системи

вимагають частих регулярних резервних копій і постійних інкрементних резервних копій.

З появою Інтернету та ери електронної комерції OLTP-системи набули повсюдного поширення. Їх можна знайти майже в кожній галузі або на кожному вертикальному ринку, а також у багатьох системах, орієнтованих на споживача. Повсякденні приклади OLTP-систем включають наступне:

- Банкомати (це класичний приклад, який найчастіше згадується) і банківські додатки в Інтернеті
- Обробка платежів за допомогою кредитних карток (як в Інтернеті, так і в магазині)
- Введення замовлень (роздрібна торгівля та бек-офіс)
- Онлайн-бронювання (продаж квитків, системи резервування тощо)
- Ведення документації (включаючи медичні записи, контроль запасів, планування виробництва, обробку претензій, продаж квитків для обслуговування клієнтів та багато інших додатків)

Як працює OLAP:

OLAP (від англ. Online Analytical Processing) - це програмне забезпечення для виконання багатовимірного аналізу на високій швидкості великих обсягів даних зі сховища даних, вітрини даних або іншого уніфікованого, централізованого сховища даних.

Більшість бізнес-даних мають кілька вимірів - кілька категорій, на які дані розбиваються для презентації, відстеження або аналізу. Наприклад, дані про продажі можуть мати кілька вимірів, пов'язаних з місцем розташування (регіон, країна, штат/область, магазин), часом (рік, місяць, тиждень, день), продуктом (одяг, чоловічий/жіночий/дитячий, бренд, тип) тощо.

Але у сховищі даних набори даних зберігаються в таблицях, кожна з яких може організовувати дані лише у двох з цих вимірів одночасно. OLAP

витягує дані з декількох реляційних наборів даних і реорганізовує їх у багатовимірний формат, який забезпечує дуже швидку обробку і дуже глибокий аналіз.

Ядром більшості OLAP-систем є OLAP-куб - багатовимірна база даних на основі масивів, яка дає змогу обробляти й аналізувати різні виміри даних набагато швидше й ефективніше, ніж традиційна реляційна база даних.

Таблиця реляційної бази даних структурована як електронна таблиця, що зберігає окремі записи у двовимірному форматі "рядок за стовпцем". Кожен "факт" у базі даних знаходиться на перетині двох вимірів - рядка і стовпця, наприклад, регіон і загальний обсяг продажів.

SQL та інструменти звітності для реляційних баз даних, безумовно, можуть запитувати, створювати звіти та аналізувати багатовимірні дані, що зберігаються в таблицях, але продуктивність сповільнюється зі збільшенням обсягів даних. А реорганізація результатів, щоб зосередитися на різних вимірах, вимагає багато роботи.

Саме тут на допомогу приходить OLAP-куб. Куб OLAP розширює єдину таблицю додатковими шарами, кожен з яких додає додаткові виміри - зазвичай наступний рівень в "ієрархії понять" виміру. Наприклад, верхній рівень куба може організовувати продажі за регіонами; додатковими рівнями можуть бути країна, штат/область, місто і навіть конкретний магазин.

Теоретично куб може містити нескінченну кількість шарів. (Куб OLAP, що представляє більше трьох вимірів, іноді називають гіперкубом). А всередині шарів можуть існувати менші кубики - наприклад, кожен шар магазину може містити кубики, що впорядковують продажі за продавцями та товарами. На практиці аналітики даних створюють OLAP-куби, що містять саме ті шари, які їм потрібні, для оптимального аналізу та продуктивності.

Drill-down - операція деталізації перетворює менш детальні дані на більш детальні за допомогою одного з двох методів - переміщення вниз в

ієрархії понять або додавання нового виміру до куба. Наприклад, якщо ви переглядаєте дані про продажі за календарний або фінансовий квартал організації, ви можете деталізувати, щоб побачити продажі за кожен місяць, рухаючись вниз в ієрархії концептів виміру "час".

Roll up - Функція згортання є протилежною до функції деталізації - вона агрегує дані в OLAP-кубі, переміщуючись вгору в ієрархії концептів або зменшуючи кількість вимірів. Наприклад, ви можете піднятися вгору в ієрархії концептів виміру "місцезнаходження", переглядаючи дані по кожній країні, а не по кожному місту.

Slice and dice - Операція зрізу створює підкуб шляхом вибору одного виміру з основного OLAP-куба. Наприклад, ви можете виконати зріз, виділивши всі дані за перший фінансовий або календарний квартал організації (часовий вимір).

Операція "кубики" ізолює підкуб, виділяючи кілька вимірів у головному кубі OLAP. Наприклад, ви можете виконати операцію зрізу, виділивши всі дані за календарними або фінансовими кварталами організації (часовий вимір) і в межах США та Канади (вимір місцезнаходження).

Pivot - Функція зведення обертає поточне подання куба, щоб відобразити нове представлення даних, що уможливлює динамічне багатовимірне представлення даних. Функція зведених таблиць OLAP порівнянна з функцією зведених таблиць в електронних таблицях, таких як Microsoft Excel, але в той час як зведені таблиці в Excel можуть бути складними, зведені таблиці OLAP відносно простіші у використанні (потрібно менше досвіду) і мають швидший час відгуку і продуктивність запитів.

OLTP проти OLAP:

Онлайнова обробка транзакцій, або OLTP, відноситься до методів обробки даних і програмного забезпечення, орієнтованих на транзакційно-орієнтовані дані і додатки.

Основна відмінність між OLAP і OLTP полягає в назві: OLAP є аналітичним за своєю природою, а OLTP - транзакційним.

Інструменти OLAP призначені для багатовимірного аналізу даних у сховищі даних, яке містить як транзакційні, так і історичні дані. Фактично, OLAP-сервер, як правило, є середнім, аналітичним рівнем рішення для сховища даних. Серед поширених застосувань OLAP - інтелектуальний аналіз даних та інші програми бізнес-аналітики, складні аналітичні розрахунки та прогнозні сценарії, а також функції бізнес-звітності, такі як фінансовий аналіз, бюджетування та прогнозне планування.

OLTP призначена для підтримки транзакційно-орієнтованих додатків, обробляючи останні транзакції якомога швидше і точніше. Серед поширених застосувань OLTP - банкомати, програмне забезпечення для електронної комерції, обробка платежів за допомогою кредитних карток, онлайн-бронювання, системи резервування та інструменти для ведення обліку.

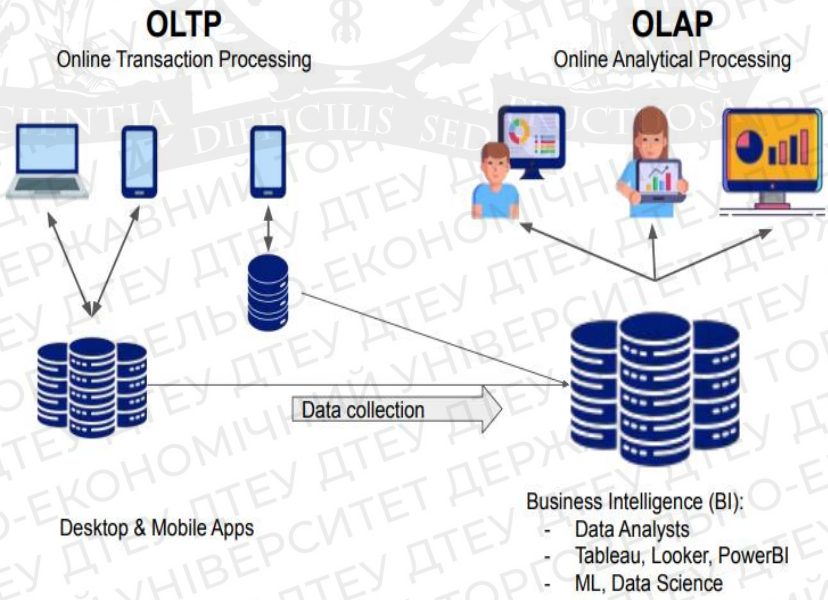

Рис. 2.6. Приклад архітектури роботи OLAP з OLTP

2.5. Висновки до розділу 2

У ході дослідження різних аспектів OLAP (Online Analytical Processing) та OLTP (Online Transaction Processing) баз даних, було ретельно проаналізовано їхні плюси та мінуси, а також здійснено порівняльний аналіз обох підходів. Дані дослідження дозволили отримати глибше розуміння сутності цих двох типів баз даних і їхнього впливу на різні аспекти сучасних програмних систем.

OLAP бази даних відіграють ключову роль у проведенні аналітики та здійсненні стратегічних рішень в організаціях. Вони дозволяють ефективно опрацьовувати великі обсяги даних, забезпечуючи можливість швидкого виконання складних аналітичних запитів. OLAP системи володіють високою продуктивністю для операцій агрегації та групування даних, що робить їх незамінними при роботі з бізнес-інтелектом. Однак їхній недолік полягає у тому, що вони не підходять для операцій вставки, оновлення та видалення даних у режимі реального часу, що робить їх менш ефективними для транзакційно-орієнтованих задач.

З іншого боку, OLTP бази даних є незамінним інструментом для обробки транзакцій та забезпечення консистентності даних в операційних середовищах. Вони гарантують високу швидкість операцій вставки, оновлення та видалення, що робить їх найкращим вибором для систем, де важлива надійність та точність даних. Однак, OLTP системи можуть стикатися з обмеженнями продуктивності під час виконання складних аналітичних запитів через їхню оптимізацію для операцій в реальному часі.

Порівнюючи обидва типи баз даних, можна зробити висновок, що вони мають різні цільові напрямки. OLAP бази даних підходять для аналітики та висновків з даних, допомагаючи робити стратегічні рішення на основі об'ємних даних. З іншого боку, OLTP бази даних спрямовані на підтримку

операційних процесів, де критично важливо забезпечення надійності, цілісності та доступності даних у режимі реального часу.

У сучасному інформаційному середовищі часто виникає необхідність поєднувати обидва підходи, створюючи гібридні рішення, які комбінують переваги OLAP та OLTP систем. Це дозволяє досягти балансу між потребами аналітики та операцій та забезпечити ефективне функціонування інформаційних систем в умовах сучасного бізнесу.

У підсумку, обидва типи баз даних мають свою важливу роль у сфері інформаційних технологій, і їхнє правильне застосування залежить від конкретних вимог та цілей організації.

РОЗДІЛ 3
РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ СИСТЕМИ ЗБОРУ, АНАЛІЗУ ТА ЗБЕРІГАННЯ МАРКЕТИНГОВИХ ДАНИХ З СОЦІАЛЬНИХ МЕРЕЖ

3.1. Архітектура системи збору, аналізу та зберігання маркетингових даних з соціальних мереж

Архітектурна конфігурація системи виглядає наступним чином: запускаємо таблицю на PostgreSQL, розгорнуту на сервері Google Compute Engine. В цій базі даних зберігаються дані клієнтів у вигляді таблиць. Для кожного запису в цій таблиці ви визиваєте Cloud Function, яка надсилає дані з API до BigQuery для подальшої обробки.

Коли клієнтська програма чи додаток робить запис в таблицю в базі даних PostgreSQL, це викликає тригер для запуску нашої Cloud Function. Ця функція діє як проміжний шар, що відповідає за отримання даних з API, обробку та передачу їх до BigQuery.

Cloud Function, після виклику, ініціює взаємодію з віддаленим API, витягаючи додаткові дані. Ця інформація може бути використана для збагачення даних або подальшої фільтрації перед їх передачею в BigQuery.

Отримавши всі необхідні дані, Cloud Function готує їх для відправлення в BigQuery. Це може включати обробку, перетворення та нормалізацію даних відповідно до схеми, встановленої у BigQuery.

Після підготовки, дані передаються в BigQuery для подальшого зберігання та аналізу. У BigQuery можна виконувати різноманітні запити та

аналітичні операції, використовуючи потужність обчислювальних ресурсів платформи.Весь описаний процес, включаючи запис даних у базу даних PostgreSQL, виклик Cloud Function, обробку та передачу даних до BigQuery, автоматизовано запускається на віртуальному сервері Compute Engine за допомогою планувальника cron. Планувальник cron визначає графік інтервалів, коли має виконуватись весь цей процес, і автоматично запускає необхідні кроки з визначеною регулярністю, забезпечуючи автоматизовану та надійну обробку даних.

Ця архітектурна конфігурація дозволяє автоматизовано збирати, обробляти та зберігати дані з бази даних PostgreSQL у BigQuery для подальшої аналітики та виконання різноманітних запитів. Перехід від транзакційної бази даних до аналітичної платформи забезпечує швидке та зручне аналізування великих обсягів даних.

Архітектурна конфігурація базується на розподіленому підході до обробки даних, що дозволяє ефективно використовувати ресурси та платформи для різних завдань. Для кожної частини системи відповідно налаштовуються ресурси та функціональність, забезпечуючи оптимальну продуктивність.

3.2. Програмна реалізація системи клієнтських даних для авторизації маркетингових даних з соціальних мереж

Створюємо робоче середовище для обробки клієнтських даних, використовуючи платформу Google Cloud. Наша конфігурація базується на використанні Docker контейнера, який містить базу даних PostgreSQL разом з клієнтськими даними. Створюємо папку “docker_build” де буде відбуватись створення Docker контейнера, це ініціалізуємо “docker-compose.yam l” Це середовище потрібне для побудови усієї системи яка буде функціонувати на сервері і запускати інші інструменти.

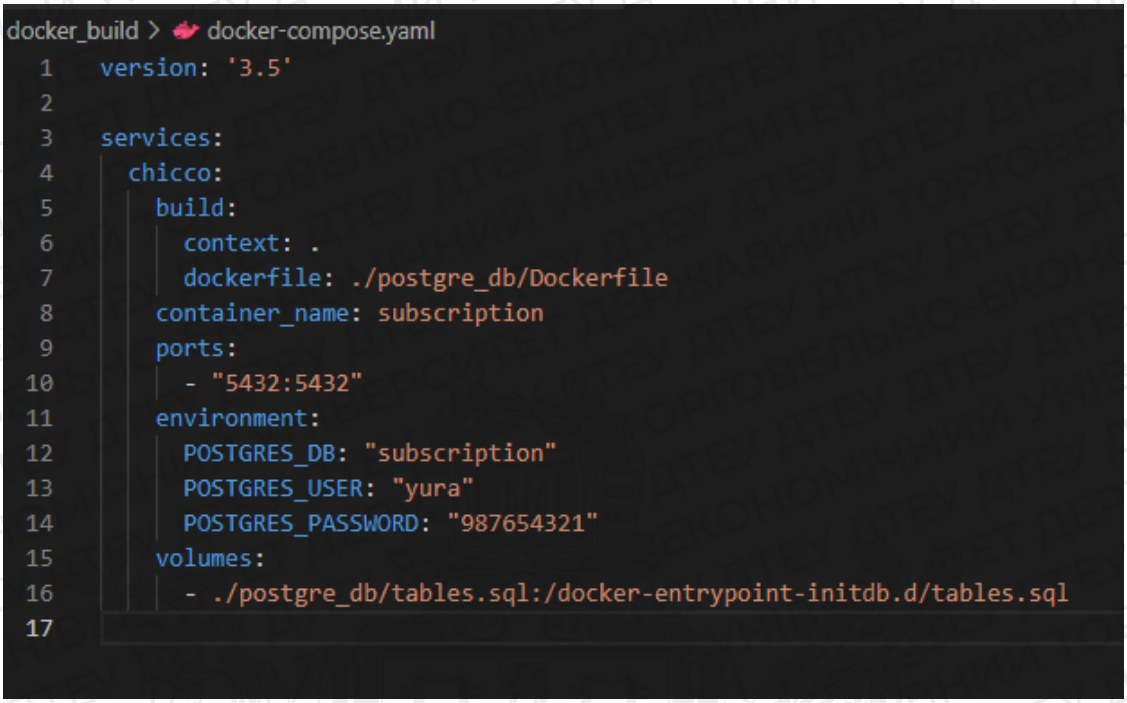

Рис. 3.1. Створення PostgreSQL на Docker контейнері

Цей файл docker-compose.yaml описує конфігурацію для запуску Docker контейнера з базою даних PostgreSQL за допомогою інструмента Docker Compose. Давайте розглянемо деталі цього файлу:

- version: '3.5': Визначає версію синтаксису Docker Compose. У вашому випадку, використовується версія 3.5.
- services: Розділ, в якому описується список контейнерів (служб), які будуть створені.
- chicco:: Назва для контейнера або служби.
- build:: Конфігурація для побудови образу контейнера.
- context: .: Вказує шлях до директорії, з якої буде виконана побудова образу. . вказує на поточну директорію (де знаходиться docker-compose.yaml).
- environment:: Задання змінних середовища для контейнера.
- POSTGRES_DB: "subscription": Задання назви бази даних PostgreSQL.

- POSTGRES_USER: "yura": Задання користувача PostgreSQL.
- POSTGRES_PASSWORD: "987654321": Задання пароля для користувача PostgreSQL.
- dockerfile: ./postgre_db/Dockerfile: Вказує шлях до Dockerfile, який буде використовуватись для побудови образу.

До цього білду додаємо SQL-файл містить команду для створення таблиці subscription у базі даних PostgreSQL. Давайте розглянемо структуру таблиці та її поля:

- id SERIAL: Це поле встановлюється як серійний (автоінкрементний) ідентифікатор. Кожен новий рядок у таблиці буде мати унікальне значення, яке автоматично збільшується з кожним додаванням нового рядка.
- name TEXT: Це текстове поле, яке зберігатиме ім'я.
- date_in TEXT: Текстове поле для збереження дати початку.
- date_out TEXT: Текстове поле для збереження дати завершення.
- bigquery_tab TEXT: Текстове поле для збереження інформації про таблицю в BigQuery.
- description TEXT NULL: Текстове поле для опису підписки. NULL означає, що це поле може мати відсутнє значення.

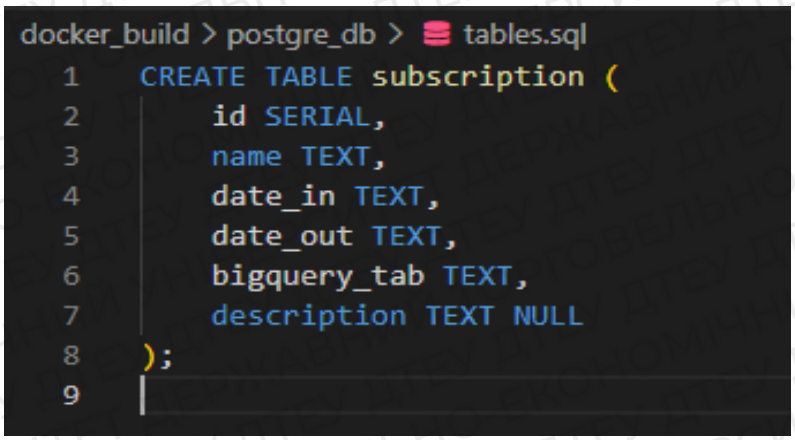

Рис. 3.2. Створення таблиці в PostgreSQL на Docker контейнері

Коли використовується цей SQL-файл під час білду бази даних PostgreSQL, ви включаєте його у Docker контейнер у директорію /docker-entrypoint-initdb.d/. Під час запуску контейнера, PostgreSQL автоматично визначає всі файли з цієї директорії та виконує їхні SQL-команди для ініціалізації бази даних.

Цей docker-compose.yaml файл описує налаштування для створення Docker контейнера з PostgreSQL базою даних, яка містить клієнтські дані. Контейнер буде доступний на порті 5432 хостової машини, і база даних буде налаштована з вказаними назвою, користувачем та паролем. Крім того, файл tables.sql виконується при старті контейнера для ініціалізації бази даних.

Нам вдалося створити Docker-контейнер, який об'єднує образ PostgreSQL з налаштуваннями та даними клієнтів. По завершенні цього етапу, відправляємо цей Docker-образ на віддалений репозиторій, такий як Docker Hub, де він може бути зручно зберігатись та використовуватись.

Після цього створюємо віртуальну машину на Google Compute Engine, вибравши підходящий образ віртуальної машини та налаштувавши його з підтримкою Docker. Це дає змогу далі розгортати наш Docker-контейнер з PostgreSQL та даними на обраному середовищі.

Тепер, коли маємо працюючу віртуальну машину з Docker, можна легко запускати наш контейнер з базою даних. Це відкриває можливість взаємодіяти з даними клієнтів, виконувати запити та здійснювати операції зберігання та аналізу даних.

3.3. Програмна реалізація системи збору, аналізу та зберігання маркетингових даних з соціальних мереж

Тепер для подальшої розробки системи робимо окремий Python файл. Цей окремий Python файл, в якому зберігаються змінні з токеном доступу та

даними для підключення до бази даних, є важливим компонентом програми або додатка. Давайте детально розглянемо, навіщо цей файл потрібний:

- Зберігання конфіденційних даних: Файл служить для зберігання конфіденційних або приватних даних, таких як токени доступу та дані для підключення до бази даних. Зберігання цих даних окремо допомагає забезпечити їх безпеку та унеможливити ненавмисний або несанкціонований доступ до них.
- Зручність та модульність: Розділення конфіденційних даних у окремий файл робить код більш зручним та модульним. Можна легко оновлювати цей файл, не втручаючись у вихідний код програми. Це особливо корисно при роботі у команді, де кожен розробник може мати свій власний файл зі своїми налаштуваннями.
- Зменшення ризику помилок: Використання окремого файлу для конфіденційних даних допомагає зменшити ризик помилок. Можна перевірити та валідувати ці дані в одному місці, а не розсипані по всьому коду.
- Безпека та захист від витоку даних: Важливо забезпечити, щоб цей файл був належним чином захищений. Треба ігнорувати його у системі контролю версій (наприклад, використовуючи .gitignore), а також зашифрувати або використовувати відповідні механізми захисту, щоб уникнути витоку конфіденційної інформації.
- Підтримка різних середовищ: Можливість легко переключати дані між різними середовищами (наприклад, робоче середовище, тестове середовище, середовище розробки) шляхом заміни цього файлу зі змінними.
- Спрощення розгортання та керування налаштуваннями: Під час розгортання додатка в продуктивне середовище або міграції на інший

сервер, можна легко встановити правильні дані підключення та інші конфігураційні параметри, змінивши лише цей окремий файл.

Рис. 3.3. Створення файлу з ключами

Цей файл з конфіденційними даними є важливим елементом розробки програмного забезпечення, який сприяє безпеці, зручності та кращому управлінню конфігурацією вашої програми.

Створення власного API:

Створюємо APi за допомогою бібліотек SQLAlchemy, Pandas та імпортуємо файл з нашими ключами. Використовуємо декілька бібліотек для створення API для підключення до бази даних PostgreSQL, тому що маємо триматися принципу лаконічності, та принципу нічого зайвого.

1. SQLAlchemy: Ми використовуємо бібліотеку SQLAlchemy для роботи з базою даних PostgreSQL через ORM (Об'єктно-Реляційне Відображення). SQLAlchemy надає зручний і програмістичний спосіб взаємодії з базою даних, дозволяючи вам працювати з об'єктами Python замість написання SQL-запитів безпосередньо. Це полегшує розробку і зберігає час.

2. Pandas: Бібліотека Pandas є потужним інструментом для роботи з даними в Python. Вона дозволяє легко читати дані з бази даних PostgreSQL, а також обробляти та аналізувати їх. Використання Pandas може значно спростити роботу з отриманими даними і дозволяє зручно використовувати їх у вашому API.
3. credfile: Ви використовуєте цю бібліотеку для отримання конфіденційних даних (логіну, паролю, тощо) для підключення до бази даних з безпечного місця. Це хороша практика для зберігання конфіденційних інформації поза кодом, щоб забезпечити безпеку даних.

```python
import sqlalchemy
import pandas as pd
from credfile import credentials


class DatabaseAPI:
    def __init__(self,):
        self.db_user = credentials.get("db_user")
        self.db_password = credentials.get("db_password")
        self.db_host = credentials.get("db_host")
        self.db_port = credentials.get("db_port")
        self.db_name = credentials.get("db_name")
        self.engine = None
```

Рис. 3.4. Ініціалізування класу

Загальна ідея полягає в тому, що ви використовуєте SQLAlchemy для забезпечення зручного способу підключення та взаємодії з базою даних PostgreSQL. Пандас використовується для опрацювання та аналізу даних, які ви отримуєте з бази даних. Крім того, ви дотримуєтеся безпеки, зберігаючи конфіденційні дані в окремому файлі за допомогою бібліотеки credfile.

Після того як було обрано і ініціалізовано клас, продовжуємо розробку API. Описуючи підключення та відключення від БД.

```python
def connect(self):
    try:
        connection = f'postgresql://{self.db_user}:{self.db_password}@{self.db_host}:{self.db_port}
        self.engine = sqlalchemy.create_engine(connection)
        print("Успешное подключение к базе данных PostgreSQL")
    except Exception as error:
        print("Ошибка при подключении к базе данных PostgreSQL:", error)


def disconnect(self):
    try:
        self.engine.dispose()
        print("Подключение к базе данных PostgreSQL закрыто")
    except Exception as error:
        print("Ошибка при закрытии подключения к базе данных PostgreSQL:", error)
```

Рис. 3.5. Підключення та відключення до БД

Ці функції connect і disconnect реалізовані в вашому класі DatabaseAPI і використовуються для керування підключенням до бази даних PostgreSQL. Ось як вони працюють:

1. **Фунцкія connect:**

Функція connect призначена для встановлення підключення до бази даних PostgreSQL.

У спробі (try) функція створює рядок підключення до бази даних за допомогою змінних, які містять інформацію про користувача (self.db_user), пароль (self.db_password), хост (self.db_host), порт (self.db_port) та назву бази даних (self.db_name).

Після створення рядка підключення функція використовує бібліотеку sqlalchemy для створення об'єкта engine, який представляє з'єднання з базою даних.

Якщо підключення вдалося встановити, виводиться повідомлення "Успешное подключение к базе данных PostgreSQL". У протилежному випадку, якщо сталася помилка (Exception), виводиться повідомлення про помилку.

2. **Фунцкія disconnect:**

Функція disconnect призначена для закриття підключення до бази даних PostgreSQL.

У спробі (try) функція викликає метод dispose() на об'єкті engine, який відповідає за підключення. Це закриває з'єднання та звільняє ресурси.

Після успішного закриття підключення виводиться повідомлення "Подключение к базе данных PostgreSQL закрыто". У разі виникнення помилки при закритті підключення, виводиться повідомлення про помилку.

```
def table_appending(self, table, data):
    try:
        data.to_sql(table, self.engine, if_exists='append', index=False)
        print("Данные успешно записаны в таблицу")
    except Exception as error:
        print("Ошибка при записи данных в таблицу:", error)

    self.disconnect()


def table_reader(self, table):
    try:
        query = f"SELECT * FROM {table}"
        data = pd.read_sql(query, self.engine)
        print("Данные успешно прочитаны")
    except Exception as error:
        print("Ошибка при чтении данных в таблице:", error)

    self.disconnect()

    return data
```

Рис. 3.6. Читання та дозапис даних до БД

3. **Фунцкія table_appending:**

Функція table_appending призначена для додавання даних (data) до таблиці (table) бази даних PostgreSQL.

У спробі (try) функція використовує метод to_sql бібліотеки Pandas для запису даних в таблицю бази даних. Вона використовує об'єкт engine, який був створений при підключенні до бази даних.

Параметр if_exists вказує, що робити, якщо таблиця вже існує. У вас вказано 'append', що означає додавання даних до існуючої таблиці, якщо вона вже існує.

Якщо операція запису в таблицю виконана успішно, виводиться повідомлення "Данные успешно записаны в таблицу".

У разі виникнення помилки під час запису даних в таблицю, виводиться повідомлення про помилку.

Після завершення операції запису функція викликає метод disconnect, щоб коректно закрити підключення до бази даних PostgreSQL.

4. **Фунцкія table_reader:**

Функція table_reader призначена для читання даних з таблиці бази даних PostgreSQL.

У спробі (try) функція формує запит SQL для вибору всіх даних з таблиці (table) і використовує метод read_sql бібліотеки Pandas для виконання запиту та отримання результатів.

Після успішного читання даних виводиться повідомлення "Данные успешно прочитаны".

У разі виникнення помилки під час читання даних з таблиці, виводиться повідомлення про помилку.

Після завершення операції читання функція викликає метод disconnect, щоб коректно закрити підключення до бази даних PostgreSQL.

В кінці функція повертає отримані дані у вигляді об'єкту Pandas DataFrame для подальшого використання.

```python
def clearing_table(self, table):
    metadata = sqlalchemy.MetaData(bind=self.engine)
    table = sqlalchemy.Table(table, metadata, autoload=True)
    delete_stmt = table.delete()

    with self.engine.begin() as conn:
        conn.execute(delete_stmt)
```

Рис. 3.7. Видалення даних з БД

5. **Фунцкія clearing_table:**

Спочатку створюється об'єкт metadata за допомогою sqlalchemy.MetaData та вказується об'єкт bind як ваш self.engine. Це дозволяє бібліотеці SQLAlchemy отримати метадані про вашу базу даних, включаючи інформацію про таблиці.

Створюється об'єкт таблиці, який представляє таблицю, яку ви хочете очистити. Ви використовуєте sqlalchemy.Table з назвою таблиці і метаданими, щоб завантажити інформацію про таблицю з бази даних (autoload=True).

Після створення об'єкта таблиці створюється SQL-запит для видалення всіх записів з таблиці за допомогою table.delete().

Використовуючи with self.engine.begin() as conn, ми відкриваємо транзакцію з базою даних, і потім виконуємо запит на видалення conn.execute(delete_stmt) всіх записів з таблиці.

3.4. CloudFunction

Цей розділ зосереджений на вичерпному огляді процесу розгортання Cloud Function на високопродуктивній платформі Google Cloud Platform (GCP) з метою створення ефективного Python API. Весь цей процес включає в себе

кілька ключових етапів, починаючи від стратегічного вибору технологій і закінчуючи останніми кроками деплою та тестування.

На початку розділу розглядається процес вибору технологій, де розглядаються різні інструменти та бібліотеки, що можуть бути використані для створення Python API. Аналізуються їхні переваги та недоліки, враховуючи вимоги проекту та ефективність взаємодії з сервісом аналітики.

Після вибору оптимальних технологій детально розглядається процес розробки API, зокрема створення логіки взаємодії з сервісом аналітики. Це включає в себе розробку функцій, які обробляють запити та відповідають на них, а також встановлення зв'язку з необхідними сервісами та інфраструктурою.

Далі висвітлюється процес конфігурації та налаштування Cloud Function на платформі GCP. Це включає в себе встановлення параметрів середовища виконання, налаштування доступу та автентифікації, а також оптимізацію ресурсів для забезпечення ефективності та швидкості виконання API.

Після завершення розробки та налаштування розглядається процес деплою, або розгортання, створеного Python API на платформі GCP. Це може включати в себе використання інструментів командного рядка або веб-інтерфейсу GCP для публікації функцій у хмарному середовищі.

Загальною метою цього розділу є надання читачеві всебічного розуміння процесу від вибору технологій до успішного розгортання та тестування Python API на платформі Google Cloud Platform для ефективної інтеграції з сервісом аналітики.

На першому етапі розглядається вступ до процесу розгортання Cloud Function на платформі Google Cloud Platform (GCP) для створення Python API, яке взаємодіє з сервісом аналітики. Цей етап є ключовим у контексті розуміння мети та значущості впровадження хмарної функції для створення веб-сервісу.

На сучасному етапі розвитку інформаційних технологій динаміка бізнес-середовища вимагає швидкого та ефективного впровадження інноваційних технологій для оптимізації бізнес-процесів. У цьому контексті використання хмарних технологій, зокрема Cloud Function на GCP, стає стратегічною перевагою.

Розгортання Python API через хмарні функції відкриває нові можливості для швидкого та масштабованого розвитку веб-сервісів. Це забезпечує гнучкість, легкість управління ресурсами та можливість концентрації на розробці функціональності, замість адміністрування інфраструктури.

У цьому вступі будуть визначені ключові аспекти, які будуть виведені на передній план у подальших розділах, такі як вибір технологій, ефективність роботи з сервісами GCP та загальний вплив розгортання хмарних функцій на стратегічний розвиток IT-інфраструктури підприємства. Також, буде розкрито необхідність цього підходу в контексті актуальних тенденцій у галузі хмарних технологій та розробки веб-сервісів.

Створення Cloud Function:

На цьому етапі детально розглядається процес розгортання Cloud Function на платформі Google Cloud Platform (GCP) для створення Python API з використанням HTTP тригера, який буде здатний взаємодіяти з віддаленим сервісом аналітики. Використовуючи технології GCP та інструменти розроблення, реалізується ефективний веб-сервіс, доступний для виклику через стандартні HTTP-запити.

Вибір мови програмування Python виявився вдалим для реалізації API, оскільки вона забезпечує легку читабельність коду та широкі можливості розробки. Визначення функціональності Cloud Function.

Далі, на етапі конфігурації Cloud Function, буде визначено ім'я функції, а також обсяги ресурсів, необхідні для ефективної роботи. Важливим аспектом

є вибір HTTP тригера, який забезпечить доступність функції через HTTP-запити, завдяки якому, ми можемо запускати цю функцію з будь-якого місця отримавши токен доступу. На етапі конфігурації HTTP тригера необхідно вказати метод HTTP-запиту, який спрацьовуватиме тригер. Можна вибрати методи POST, GET, або інші, залежно від потреб веб-сервісу. Також, вказати можна URL-шлях, за яким буде доступна функція.

Розгортання Cloud Function:

Процес розгортання Cloud Function на платформі Google Cloud Platform для створення Python API із використанням тригера, що надає можливість взаємодії з віддаленим сервісом аналітики. Розгортання функції є ключовим кроком у створенні доступного та ефективного веб-сервісу, тому розробник детально розглядає всі аспекти цього процесу.

Після написання коду розробник виконує конфігурацію самої Cloud Function, обираючи відповідні параметри, такі як ім'я функції та обсяги ресурсів. Розробник також визначає HTTP тригер, що дозволяє функції реагувати на HTTP-запити. Для забезпечення правильної роботи функції, вказуємо необхідні залежності, які включають бібліотеки та інші компоненти, необхідні для коректної роботи веб-сервісу. Ці залежності додаватимуться до конфігураційного файлу функції.

Окрему увагу приділяється конфігурації обсягів ресурсів функції. Максимальний час виконання встановлюється на 540 секунд, забезпечуючи адекватний час для взаємодії із заданим сервісом аналітики. Обсяг оперативної пам'яті встановлюється на 256 МБ для забезпечення необхідних ресурсів для виконання завдань функції.

Після завершення цих етапів, можна використовувати інструменти Google Cloud SDK для деплою функції на платформу GCP. Наступним кроком

є проведення тестування функції за допомогою HTTP-запитів, щоб переконатися в її правильній роботі.

Такий підхід до розгортання веб-сервісу з використанням Cloud Function та HTTP тригера дозволяє створити легкодоступний та ефективний інтерфейс, який здатний обробляти HTTP-запити для взаємодії із заданим сервісом аналітики в межах встановленого часу та обсягу пам'яті.

3.5. Висновки до розділу 3

В розділі 3 дипломної роботи була проведена ретельна розробка програмного забезпечення для системи збору, аналізу та зберігання маркетингових даних з соціальних мереж. В даному висновку відзначимо ключові аспекти, що були висвітлені у кожному підрозділі цього розділу.

У підрозділі 3.1 "Архітектура системи збору, аналізу та зберігання маркетингових даних з соціальних мереж" була розглянута загальна структура системи. Розгорнута архітектура дозволяє ефективно взаємодіяти з різними соціальними мережами, забезпечуючи надійний та швидкий збір і обробку даних.

У підрозділі 3.2 "Програмна реалізація системи клієнтських даних для авторизації маркетингових даних з соціальних мереж" розглянута реалізація механізму авторизації, який забезпечує безпеку та конфіденційність отриманих даних. Використані сучасні методи шифрування та протоколи забезпечують захист інформації від несанкціонованого доступу.

Підрозділ 3.3 "Програмна реалізація системи збору, аналізу та зберігання маркетингових даних з соціальних мереж" висвітлює ключові етапи збору та обробки даних. Розроблені алгоритми аналізу дозволяють отримувати цінну інформацію для подальшого використання в маркетингових стратегіях.

У підрозділі 3.4 "CloudFunction" розглянута інтеграція з хмарним сервісом, що полегшує масштабування та забезпечує стабільну роботу

системи в умовах великого обсягу даних. Використання CloudFunction сприяє оптимізації ресурсів та забезпечує високу доступність системи.

Отже, в цьому розділі дипломної роботи були детально розглянуті аспекти архітектури системи, програмної реалізації авторизації та збору даних з соціальних мереж, а також інтеграції з хмарним сервісом. Результатом роботи є створена ефективна система, яка відповідає сучасним вимогам маркетингового аналізу та може служити ефективним інструментом для бізнесу.

ВИСНОВКИ ТА ПРОПОЗИЦІЇ

У даній дипломній роботі було проведено глибоке теоретичне дослідження механізмів автоматизованої системи збору, аналізу та зберігання маркетингових даних, зокрема в контексті їх отримання з соціальних мереж. У першому розділі роботи висвітлено поняття та теоретичні основи збору маркетингових даних з соціальних мереж, а також проведено аналіз відомих методів систем збору, аналізу та зберігання даних, надаючи читачеві зрозуміле уявлення про актуальні тенденції в даній області. Крім того, в розділі визначена характеристика існуючих методів та програмних засобів збору, аналізу та зберігання маркетингових даних.

Другий розділ дипломної роботи присвячений проектуванню системи збору, аналізу та зберігання маркетингових даних. У цьому розділі було проведено докладний аналіз і розглянуті ключові аспекти, які визначають успішність та ефективність системи.

Початковим етапом було визначено цілі та завдання системи. Це включало в себе розуміння вимог до обсягу та типів маркетингових даних, їхнього обсягу та структури. Також були визначені очікувані результати впровадження системи, такі як покращення часу реакції на зміни в маркетингових стратегіях, підвищення точності аналізу даних та забезпечення надійності системи.

Детально був розглянутий вибір між ETL (Extract, Transform, Load) та ELT (Extract, Load, Transform) підходами до обробки даних. Для кожного методу були проаналізовані переваги та недоліки з точки зору специфіки маркетингових даних, їхнього обсягу та структури.

Досліджено використання технології MPP для оптимізації обробки даних в режимі реального часу. Було визначено, як цей підхід може покращити швидкість та масштабованість системи збору та аналізу маркетингових даних.

Опрацьовано питання вибору між транзакційною (OLTP) та аналітичною (OLAP) обробкою даних. Проаналізовані сценарії використання для обох методів, зосереджуючись на їхній відповідності до потреб збору та аналізу маркетингових даних.

Результатом цього розділу є глибокий аналіз та обґрунтування виборів, зроблених у процесі проектування системи збору, аналізу та зберігання маркетингових даних. Розглянуті аспекти дозволяють побудувати ефективну та високопродуктивну систему, яка задовольняє конкретні вимоги маркетингового аналізу в індустрії. Крім того, враховуючи сучасні тенденції та технологічні інновації, система буде готовою до майбутніх викликів та розширень.

Третій розділ дипломної роботи присвячений розробці програмного забезпечення для системи збору, аналізу та зберігання маркетингових даних з соціальних мереж. Цей розділ включає в себе деталізовану концепцію архітектури системи та реалізацію ключових елементів програмного комплексу.

У першому підрозділі розглядається загальна архітектура системи. Описується структура компонентів, їх взаємодія та розподілена природа системи. Приділяється увага вибору технологій, які найкраще відповідають вимогам системи, зокрема з урахуванням обраного методу ETL або ELT, технології MPP та типу баз даних.

Другий підрозділ охоплює розробку інтерфейсу для взаємодії з соціальними мережами та авторизації отримання маркетингових даних. Аналізуються протоколи авторизації, безпека передачі даних, а також взаємодія з API соціальних мереж.

Третій розділ описує реалізацію основних функцій системи, таких як збір, обробка та зберігання маркетингових даних. Аналізуються використані алгоритми, методи агрегації та оптимізації запитів для забезпечення швидкодії та ефективності обробки великих обсягів даних.

Четвертий підрозділ розглядає використання CloudFunction для оптимізації деяких частин системи. Обґрунтовується вибір хмарної платформи та пояснюється, як CloudFunction використовується для розширення функціональності та забезпечення гнучкості системи.

Реалізація цих елементів дозволяє створити повнофункціональну систему збору, аналізу та зберігання маркетингових даних з соціальних мереж. Вона відповідає вимогам, що були визначені на етапі проектування, і забезпечує ефективний інструмент для маркетологів та аналітиків для оптимізації стратегій та прийняття обґрунтованих управлінських рішень. У цілому, проведене теоретичне дослідження та практична реалізація дипломної роботи підтверджують актуальність та ефективність вибраних методів та технологій у сфері збору, аналізу та зберігання маркетингових даних з соціальних мереж. Робота відповідає сучасним вимогам та тенденціям у сфері маркетингу, дозволяючи здійснювати швидкий та точний аналіз масивів даних для прийняття обґрунтованих управлінських рішень.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. https://cloud.google.com/docs – документація на використання сервісів Google Cloud – 2023.рік
2. Fluent Python. Clear, Concise, and Effective Programming. – Luciano Ramalho – O'Reilly – 2022.рік
3. https://docs.docker.com/get-started/overview/ – документація на використання сервісів Docker – 2023.рік
4. PostgreSQL: Up and Running. – Regina Obe – O'Reilly – 2012.рік
5. https://www.postgresql.org/docs/current/ – документація на використання бази даних PostgreSQL – 2023.рік
6. Google Cloud Platform in Action – J.J. Geewax – 2018.рік
7. RESTful Web APIs: Services for a Changing World – Leonard Richardson (Author), Mike Amundsen (Author), Sam Ruby (Author) – O'Reilly – 2013.рік

ДОДАТКИ

Додаток А

Лістинг прграмного коду

postgre_api.py > DatabaseAPI > table_reader

```python
import sqlalchemy
import pandas as pd
from credfile import credentials


class DatabaseAPI:
    def __init__(self,):
        self.db_user = credentials.get("db_user")
        self.db_password = credentials.get("db_password")
        self.db_host = credentials.get("db_host")
        self.db_port = credentials.get("db_port")
        self.db_name = credentials.get("db_name")
        self.engine = None


    def connect(self):
        try:
            connection = f'postgresql://{self.db_user}:{self.db_password}@{self.db_host}:
            self.engine = sqlalchemy.create_engine(connection)
            print("Успешное подключение к базе данных PostgreSQL")
        except Exception as error:
            print("Ошибка при подключении к базе данных PostgreSQL:", error)


    def disconnect(self):
        try:
            self.engine.dispose()
            print("Подключение к базе данных PostgreSQL закрыто")
        except Exception as error:
            print("Ошибка при закрытии подключения к базе данных PostgreSQL:", error)
```

```
def table_appending(self, table, data):
    try:
        data.to_sql(table, self.engine, if_exists='append', index=False)
        print("Данные успешно записаны в таблицу")
    except Exception as error:
        print("Ошибка при записи данных в таблицу:", error)

    self.disconnect()


def table_reader(self, table):
    try:
        query = f"SELECT * FROM {table}"
        data = pd.read_sql(query, self.engine)
        print("Данные успешно прочитаны")
    except Exception as error:
        print("Ошибка при чтении данных в таблице:", error)

    self.disconnect()

    return data


def clearing_table(self, table):
    metadata = sqlalchemy.MetaData(bind=self.engine)
    table = sqlalchemy.Table(table, metadata, autoload=True)
    delete_stmt = table.delete()

    with self.engine.begin() as conn:
        conn.execute(delete_stmt)
```

```
def diploma_stat(ReportsURL, body, headers):
    body = json.dumps(body, indent=4)

    while True:
        try:
            req = requests.post(ReportsURL, body, headers=headers)
            req.encoding = 'utf-8'

            if req.status_code == 400:
                print(
                    "Параметры запроса указаны неверно или достигнут лимит отчетов в очер
                print("RequestId: {}".format(
                    req.headers.get("RequestId", False)))
                print("JSON-код запроса: {}".format(u(body)))
                print("JSON-код ответа сервера: \n{}".format(u(req.json())))
                break
            elif req.status_code == 200:
                print("Отчет создан успешно")
                print("RequestId: {}".format(
                    req.headers.get("RequestId", False)))
                print("Содержание отчета: \n{}".format(u(req.text)))
                break
            elif req.status_code == 201:
                print("Отчет успешно поставлен в очередь в режиме офлайн")
                retryIn = int(req.headers.get("retryIn", 60))
                print("Повторная отправка запроса через {} секунд".format(retryIn))
                print("RequestId: {}".format(
                    req.headers.get("RequestId", False)))
                sleep(retryIn)
            elif req.status_code == 202:
                print("Отчет формируется в режиме офлайн")
                retryIn = int(req.headers.get("retryIn", 60))
                print("Повторная отправка запроса через {} секунд".format(retryIn))
                print("RequestId:  {}".format(
                    req.headers.get("RequestId", False)))
                sleep(retryIn)
            elif req.status_code == 500:
                print(
                    "При формировании отчета произошла ошибка. Пожалуйста, попробуйте пов
                print("RequestId: {}".format(
```

MS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
def data_getter():
    params = [
        "Clicks",
        "Impressions",
        "Cost",
        "Date",
        "CampaignName",
        "AdGroupName",
        "CampaignId",
        "AdGroupId",
        "AdNetworkType",
        "AdId"
    ]

    yesterday = datetime.strftime(datetime.now() - timedelta(1), '%Y-
    print(yesterday)

    body = report_body(yesterday, params)
    data = diploma_stat(ReportsURL, body, headers).text.splitlines()
    data = [dict(zip(params, line.split("\t"))) for line in data]

    return data


def extract_data(id_val, result):
    for obj in result:
        if 'Id' in obj and obj['Id'] == int(id_val):
            keys_to_check = ['TextAd', 'TextImageAd', 'CpmBannerAdBui
            for key in keys_to_check:
                if key in obj:
                    return obj[key].get('Href')

    return None


def table_appending(table, df):
    client = bigquery.Client()
    table_id = "upheld-now-318513.diploma_ads.current_month"

    job_config = bigquery.LoadJobConfig(
```

LEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
def table_appending(table, df):
    client = bigquery.Client()
    table_id = "upheld-now-318513.diploma_ads.current_month"

    job_config = bigquery.LoadJobConfig(
        write_disposition="WRITE_APPEND"
    )

    job = client.load_table_from_dataframe(
        df, table_id, job_config=job_config
    )

    return job


def df_modifier(data):
    df = pd.DataFrame(data)

    df = df[df['AdId'] != '--']
    idis = df['AdId'].unique()

    idis = [int(x) for x in idis]

    body = ids_body(idis)
    datalinks = diploma_stat(ReportsURLads, body, headers).json()

    result = datalinks.get("result").get("Ads")

    df["Href"] = df['AdId'].apply(extract_data, result=result)

    df = df.astype({
        'Clicks': np.int64,
        'Impressions': np.int64,
        'Cost': float,
        'Date': str,
        'CampaignName': str,
        'AdGroupName': str,
        'CampaignId': str,
        'AdGroupId': str,
        'AdNetworkType': str,
        'AdId': str,
        'Href': str
```

OBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
    return df


def diploma():
    data = data_getter()

    if len(data) != 0:
        df = df_modifier(data)
        df.to_csv('data.csv', mode='w', index=False)

        table = 'diploma_set.diploma'
        table_appending(table, df)

    elif len(data) == 0:
        print('No data!!!')


if __name__ == "__main__":
    diploma()
```