Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)
Кафедра Інформатики
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

АНАЛІЗ ЗАСТОСУВАНЬ ХМАРНИХ ТЕХНОЛОГІЙ ДЛЯ ЗАДАЧ BIG DATA
(тема)

Виконав:
студент 2 курсу, групи ІНФм-22-1
Самойленко Б.В.
(прізвище, ініціали)

Спеціальності 122 Комп'ютерні науки
(код і повна назва спеціальності)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

Керівник проф. Гороховатський В.О.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ Кобилін О.А.
(підпис) (прізвище, ініціали)

2024 р.

Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)

Кафедра Інформатики
(повна назва)

Рівень вищої освіти другий (магістерський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Самойленку Богдану Вікторовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Аналіз застосувань хмарних технологій для задач big data

затверджена наказом по університету від 3 листопада 2023 року № 1280Ст

2. Термін подання студентом роботи до екзаменаційної комісії 23 грудня 2023 р.

3. Вихідні дані до роботи наукова-технічна література, методи обробки великих даних за допомогою хмарних сервісів, програмні засоби для роботи з хмарними технологіями, документація з використання хмарних сервісів

4. Перелік питань, що потрібно опрацювати в роботі

1. Дослідити особливості хмарних технологій та їх роль у роботі з big data.
2. Розглянути технології AWS для обробки великих даних.
3. Розробити конвеєр, який дозволить автоматизовано збирати дані про оцінки студентів.
4. Розробити та дослідити конвеєр даних для формування звітів.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) актуальність теми, переваги застосування хмарний технологій для обробки великих даних, порівняння хмарних провайдерів, методи обробки великих даних за допомогою хмарних сервісів, архітектура конвеєру даних для поглинання оцінок студентів, обробки помилок, імпорт оцінок студентів до сховища даних, загальна архітектура розробленого конвеєру даних, аналіз звіту про успішність студентів.

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	03.11.2023	виконано
2	Аналіз завдання, підбір літератури	04.11.23-06.11.23	виконано
3	Аналіз літератури з досліджуваної проблеми	06.11.23-15.11.23	виконано
4	Аналіз хмарних технологій в рамках big data	15.11.23-20.11.23	виконано
5	Аналіз методів обробки великих даних	20.11.23-24.11.23	виконано
6	Розробка конвеєрів даних	24.11.23-01.12.23	виконано
7	Оформлення пояснювальної записки	01.12.23-11.12.23	виконано
8	Перевірка на плагіат	12.12.2023	виконано
9	Рецензування	15.12.2023	виконано
10	Підготовка презентації та доповіді	20.12.2023	виконано
11	Занесення роботи в електронний архів	02.01.2024	виконано
12	Попередній захист кваліфікаційної роботи	02.01.2024	виконано

Дата видачі завдання 3 листопада 2023 р.

Студент _____
(підпис)

Керівник роботи _____ проф. Гороховатський В.О.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ/ABSTRACT

Пояснювальна записка до кваліфікаційної роботи: 75 с., 1 табл., 26 рис., 48 джерел.

ХМАРНІ ТЕХНОЛОГІЇ, BIG DATA, КОНВЕЄР ДАНИХ, AMAZON WEB SERVICES, ПОГЛИНАННЯ ВЕЛИКИХ ДАНИХ, ПОБУДОВА ЗВІТІВ, SERVELESS, AWS LAMBDA.

Об’єктом дослідження є хмарні технології у контексті обробки та агрегації даних для побудови звітів.

Метою дослідження є вивчення та аналіз хмарних технологій AWS у контексті їх застосування для обробки та аналізу великих обсягів даних у системі для побудови звітів про успішність студентів.

Проведено дослідження застосування хмарних технологій для опрацювання великих даних. Розглянуто процес побудови конвеєру даних та методи його реалізації за допомогою хмарних сервісів, які надає Amazon Web Services.

У результаті проведеного дослідження розроблено та досліджено конвеєри даних для поглинання інформації про успішність студентів навчальної платформи та побудови звітів на їх основі.

CLOUD COMPUTING, BIG DATA, DATA PIPELINE, AMAZON WEB SERVICES, BIG DATA INGESTION, REPORT GENERATION, SERVERLESS, AWS LAMBDA.

The object of the research is cloud technologies in the context of data processing and aggregation for building reports.

The aim of the research is to investigate and analyze AWS cloud technologies in the context of their usage to process and analyze large volumes of data in the system to build student proficiency reports.

A study of the use of cloud technologies for processing big data was conducted. The process of building a data pipeline and methods of its implementation using cloud services provided by Amazon Web Services are considered.

As a result of the conducted research, a data pipeline was developed for ingesting data about the student progress within the learning platform, also a report system was built based on it.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

AWS – Amazon Web Services

GCP – Google Cloud Platform

KDS – Kinesis Data Streams

KDF – Kinesis Data Firehose

KVS – Kinesis Video Streams

S3 – Simple Storage Service

EMR – Elastic MapReduce

ETL – Extract, Transform, Load (Витяг, Перетворення та Завантаження)

IaC – Infrastructure as code (Інфраструктура як код)

ВСТУП

У сучасному цифровому світі інформаційні технології генерують надзвичайно великий обсяг даних, з яким організації повинні працювати. Обробка великих даних є необхідною для підтримки прийняття рішень, розуміння клієнтів та оптимізації бізнес-процесів. Важливість опрацювання великих даних відзначається в різних галузях, включаючи бізнес, освіту, медицину, науку та багато інших [1-8].

Важливість використання великих даних в бізнесі полягає в можливості аналізу ринку, конкурентів та споживачів. Збирання та аналіз даних допомагає підприємствам легше розуміти, як їх продукти або послуги сприймаються споживачами. Це дає змогу покращити якість продукції та адаптувати її до потреб ринку. Крім того, великі дані дозволяють ефективно управляти запасами, прогнозувати попит і вдосконалювати стратегії маркетингу.

У медицині великі дані використовуються для аналізу даних пацієнтів та виявлення тенденцій у здоров'ї населення. Це допомагає лікарям вдосконалювати методи діагностики та лікування, а також прогнозувати розвиток захворювань.

Великі дані грають ключову роль в дослідженнях та розвитку нових технологій. Цей обсяг даних створює складнощі у управлінні та збереженні інформації. Дані надходять з різноманітних джерел, таких як соціальні медіа, IoT-пристрої, сенсори, мобільні мережі і багато інших. За даними досліджень, 2/3 всіх даних було створено за останні два роки. Наприклад, інформація від сучасних автомобілів генерує 4000 гігабайт даних щодня [1].

Традиційно компанії віддають перевагу використанню власних серверів для збереження даних. Зростання обсягу даних вимагає ускладненого керування інфраструктурою та призводить до збільшення витрат, що впливає на гнучкість підприємства. Хмарні технології надають рішення для цих проблем, запропонувавши практично нескінченні

можливості зберігання та послуги забезпечення безпеки даних. Це звільняє власників даних від необхідності витрати великих зусиль на їх опрацювання та дозволяє їм приділяти більше уваги на розробку застосунків.

Використання хмарних технологій дозволяє вирішувати складні завдання обробки та аналізу великих обсягів даних, надаючи доступ до віртуальних обчислювальних ресурсів та послуг. Хмарні платформи надають рішення для збереження даних, обробки, машинного навчання, розподіленої аналітики та глибокого навчання. Вони дозволяють компаніям і дослідницьким організаціям використовувати потужні інфраструктури та інструменти без необхідності великих витрат на обладнання та обслуговування [9-15].

У цій роботі досліджується використання хмарних технологій для освітньої платформи. Хмарні технології надають освітнім платформам можливість легко розширювати свої інфраструктури та забезпечувати безпеку даних. Ці дані створюються з різних джерел у великих обсягах та з високою швидкістю. Крім того, дані набувають різних форм, включаючи структуровані, напівструктуровані та неструктуровані дані. Ця різноманітність джерел та форм даних вимагає ефективного зберігання в відповідному сховищі. Для освітніх платформ викликом є не лише збереження, але і обробка даних з метою отримання статистичних даних або складання звітів.

1 АНАЛІЗ ТА ЗАСТОСУВАННЯ ХМАРНИХ ТЕХНОЛОГІЙ ДЛЯ BIG DATA

1.1 Визначення хмарних технологій та big data

Існує багато варіантів визначення, що таке «хмарні технології» («хмарні обчислення», «cloud computing»), але, загалом, думки науковців збігаються в тому, що хмарні технології – це модель, що передбачає доступ до обчислювальних ресурсів через Інтернет з можливістю швидкого масштабування та налаштування відповідно до потреб користувача. Ці ресурси надаються та звільняються на вимогу, з мінімальними витратами для користувача завдяки спільному набору ресурсів, які надаються провайдером хмарних послуг [1-7, 16].

Актуальність хмарних технологій в сучасному інформаційному світі є надзвичайно важливою через ряд чинників, які впливають на бізнес, науку, технології та глобальне суспільство загалом. Ключові аспекти, що підкреслюють актуальність їх аналізу та вивчення [4]:

– хмарні технології дають можливість компаніям уникнути значних витрат на придбання, налаштування та підтримку фізичного обладнання для обчислень та зберігання даних. Наприклад, для малого та середнього бізнесу це надає змогу зосередитися на розробці самого продукту, а не витрачати час на налаштування інфраструктури для програмного забезпечення;

– хмарні сервіси надають можливість миттєво масштабувати ресурси відповідно до змінних потреб, що надає змогу швидко реагувати на зміну навантаження, наприклад, якщо застосунок має певну сезонність, як освітні платформи, то в період пікових навантажень, як початок семестру, додаткові ресурси можуть бути дуже швидко додані до системи, щоб впоратись з навантаженням;

– хмарні технології надають доступ до даних та обчислювальних ресурсів з будь-якого місця, де є доступ до Інтернету. Це сприяє роботі на

віддалених робочих місцях, забезпечує спільну роботу над проектами і полегшує глобальний обмін інформацією;

– для наукових та дослідницьких організацій хмарні технології надають доступ до потужних обчислювальних ресурсів, що дозволяє проводити складні обчислення, моделювання та аналіз великих даних;

– більшість хмарних постачальників мають високий рівень захисту даних, включаючи механізми шифрування та контролю доступу.

Загалом, хмарні технології стали необхідним елементом для інноваційного росту та ефективного функціонування сучасних організацій, незалежно від їх розміру та галузі.

Основні характеристики хмарних технологій включають масштабованість, гнучкість та доступність. Користувачі можуть збільшувати або зменшувати ресурси в залежності від потреб, сплачуючи лише за використані ресурси, що робить хмарні технології ефективними для компаній будь-якого розміру, сприяючи оптимізації витрат [6].

Варто зазначити, що хмарні технології передбачають надання сервісів відповідно до певних моделей, які мають різний рівень контролю. Хмарний сервіс – це певна програма або платформа, яка надається хмарним провайдером. Найпоширеніші моделі визначено в таблиці 1.1 [8, 14].

Дані моделі відрізняються і ціноутворенням, чим менше втручання потрібно зі сторони користувача тим більше коштують послуги сервісу, тобто SaaS модель найдорожча в цьому випадку.

Вибір моделі повинен базуватися на потребах проекту та ресурсах користувача. IaaS надає більший контроль, PaaS полегшує розробку, а SaaS надає готові рішення.

Таблиця 1.1 – Порівняння моделей надання хмарних сервісів

Модель надання хмарних сервісів	**Опис**	**Переваги**	**Недоліки**	**Приклади**
IaaS (Infrastructure as a Service)	Користувачам надаються віртуальні обчислювальні ресурси, на яких можна розгортати власні операційні системи та додатки.	1. Більший контроль над сервісом. 2. Можливість налаштовувати середовище під власні потреби.	1. Вимагає більше управління. 2. Потребує експертизи у конфігуруванні та управлінні.	Amazon EC2, Google Compute Engine
PaaS (Platform as a Service)	Користувачам надається платформа для розробки та розгортання додатків, без необхідності керування інфра-структурою.	1. Швидший процес розробки додатків. 2. Менше адміністрування інфраструктури.	1. Обмежена можливість налаштування інфраструктури 2. Залежність від можливостей платформи.	AWS Elastic Beanstalk, Google App Engine, and Adobe Commerce
SaaS (Software as a Service)	Користувачі отримують доступ до готових програм та сервісів через хмару.	1. Немає необхідності у розробці та адмініструванні додатків. 2.Легкий доступ до готових рішень.	1. Обмежена можливість налаштування. 2. Залежність від постачальника SaaS.	Salesforce, Google Workspace, Looker

Що стосується big data, то цей термін описує великі, складні та різноманітні об’єми даних, що вимагають нових інструментів, аналітичних методів та інфраструктури для ефективної обробки, зберігання та аналізу [9]. Великі дані характеризуються ключовими характеристиками, які часто називають «5V» (рис. 1.1) [9]:

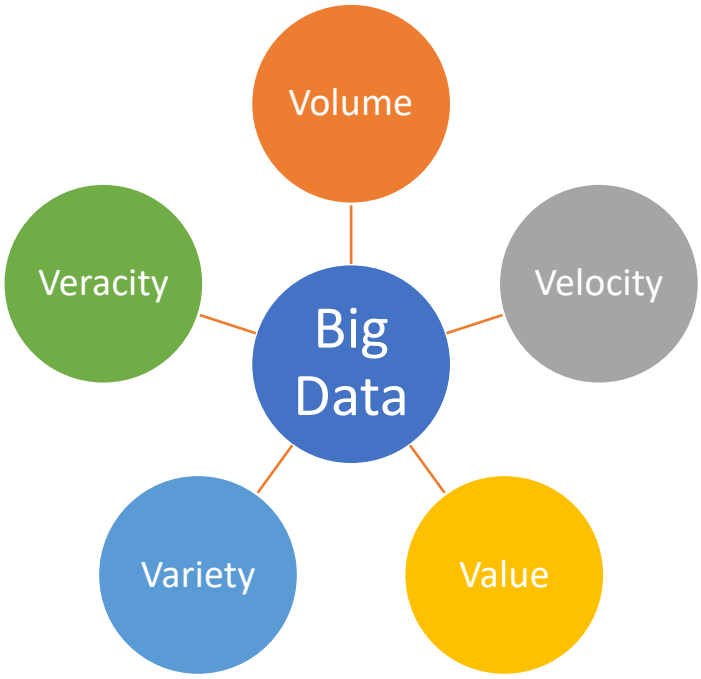

Рисунок 1.1 – «5V» Big Data

– об’єм (volume). Великі дані вказують на величезний масштаб даних, який часто вимірюється терабайтами, пентабайтами та навіть більше, і перевищує можливості традиційних методів та інструментів обробки даних;

– різноманітність (variety). Великі дані охоплюють різні типи даних, включаючи структуровані дані (такі як бази даних та таблиці), напівструктуровані дані (наприклад, файли формату JSON та XML) та неструктуровані дані (такі як текст, зображення та відео);

– швидкість (velocity). Великі дані надходять з високою швидкістю та вимагають обробки в режимі реального часу або наближеного до реального; вони генеруються неперервно з різних джерел, таких як соціальні мережі, датчики та онлайн-транзакції;

– правдоподібність (veracity) стосується якості та надійності даних. Великі дані часто походять з різних джерел із різним рівнем точності та надійності. Забезпечення правдоподібності даних є ключовим для значущого аналізу;

– цінність (value) вказує на потенційну користь та цінність, яку можна отримати з аналізу великих даних. Основною метою обробки великих даних

є виділення цінних інсайтів та інформації, що допомагає у прийнятті обґрунтованих рішень.

Ці характеристики підкреслюють виклики та можливості, пов’язані з великими даними, коли організації прагнуть збирати, зберігати, обробляти та отримувати цінні уявлення з масивів різноманітних даних.

1.2 Застосування хмарних технологій для задач big data

Хмарні технології дозволяють розв’язувати складні завдання обробки та аналізу великих обсягів даних за допомогою використання віртуальних обчислювальних ресурсів та послуг, які надають хмарні постачальники. Існує набір задач big data, які хмарні технології допомагають виконати [11]:

– зберігання даних (Data Storage). Хмарні платформи, такі як Amazon S3, Google Cloud Storage та Microsoft Azure Storage, надають масштабовані засоби для зберігання великих обсягів даних. Дані можна зберігати в хмарі без необхідності власної інфраструктури;

– обробка даних (Data Processing). Хмарні обчислювальні сервіси, такі як Amazon Elastic MapReduce (EMR) або Google Cloud Dataproc, дозволяють розподілену обробку великих обсягів даних. Це дозволяє ефективно виконувати складні обчислення та аналізувати великі набори даних;

– машинне навчання (Machine Learning). Хмарні платформи надають інструменти та сервіси для розробки, навчання та застосування моделей машинного навчання на великих обсягах даних. Amazon SageMaker, Google Cloud AI Platform та Microsoft Azure Machine Learning – це лише кілька прикладів;

– візуалізація та аналіз даних. Хмарні сервіси, такі як Tableau та Google Data Studio, дозволяють створювати інтерактивні дашборди та візуалізації для аналізу великих обсягів даних. Це допомагає зрозуміти тренди;

– розподілена аналітика (Distributed Analytics). Хмарні сервіси дозволяють використовувати розподілені бази даних та інструменти, такі як Amazon Redshift чи Google BigQuery, для виконання складних аналітичних запитів на великих наборах даних;

– аналіз тексту та обробка мови (Text Analytics and NLP). Хмарні сервіси, наприклад Amazon Comprehend або Google Cloud Natural Language, надають можливості аналізу тексту та обробки природної мови для виділення інформації з текстових даних;

– обробка стрімінгових даних (Streaming Data Processing). Для обробки стрімінгових даних, які надходять у реальному часі, можна використовувати хмарні сервіси, такі як Amazon Kinesis або Google Cloud Dataflow;

– глибоке навчання (Deep Learning). Для розв’язання завдань глибокого навчання на великих обсягах даних можна використовувати спеціалізовані хмарні платформи, які надають графічні обчислювальні ресурси.

Використання сучасних хмарних технологій для задач big data дозволяє компаніям та дослідницьким організаціям використовувати потужні інфраструктури та інструменти без необхідності великих витрат на обладнання та обслуговування.

Одним із найважливіших завдань big data для виконання якого хмарні технології грають ключову роль – є розподілена обробка даних (distributed computing). Це стратегія обробки великих обсягів інформації, де завдання розділяються на менші частини та обчислення проводяться одночасно на кількох обчислювальних ресурсах. Цей підхід є важливим в контексті великих даних з кількох причин [17].

В першу чергу, це забезпечує швидкість обробки. Завдяки паралельним обчисленням можна суттєво зменшити час, необхідний для аналізу та обробки великих обсягів даних.

Крім того, розподілена обробка дозволяє досягти масштабованості. При зростанні обсягів даних не потрібно повністю міняти обчислювальну

інфраструктуру. Замість цього, можна додавати нові ресурси для забезпечення ефективності обробки.

Цей підхід також допомагає ефективно використовувати ресурси. Замість вкладення великих коштів у один потужний сервер, розподілена обробка використовує доступні ресурси більш ефективно, розділяючи завдання між багатьма пристроями.

Крім того, важливість розподіленої обробки даних полягає в здатності проводити обробку в реальному часі. Це особливо корисно для задач, які потребують оперативного аналізу даних, таких як моніторинг та аналіз стрімінгової інформації.

Остаточно, хмарні технології стають ключовими у реалізації розподіленої обробки даних. Вони надають доступ до масштабованих обчислювальних ресурсів, що дозволяє ефективно реалізовувати стратегію розподіленої обробки великих обсягів даних.

1.3 Переваги та недоліки використання хмарних технологій для завдань big data

Використання хмарних технологій має ряд значних переваг для компаній. Назвемо переваги застосування хмарних технологій для завдань big data [15]:

– масштабованість. Хмарні технології передбачають надання безмежних обчислювальних та зберігальних ресурсів, які можна миттєво масштабувати в залежності від потреб;

– витрати. В разі використання власних обчислювальних ресурсів, налаштування необхідного обладнання – це дуже дороговартісний процес, тим паче якщо компанія хочу виконати тільки Proof of Concept, тобто підвередити практично, що конкретне завдання може бути тим чи іншим чином, то закупівля обладнання не має сенсу для цього, в цьому випадку

хмарні технології допомагають зменшити початкові витрати на проект. Також варто зазначити, що попри, малі початкові витрати, загальні витрати можуть швидко зростати, тому потрібно постійно наглядати за рівнем витрат;

– стійкість. Дані є найважливішим активом бізнесу, їх втрата може привести до великих збитків. Хмарні технології мають високий ступінь гарантії того, що дані не будуть втрачені в разі збою, завдяки процесу реплікації, який є основую високого ступеня доступність відповідних хмарних сервісів, що, в свою чергу, допомагає уникнути збитків.

Загалом, використання хмарних технологій для роботи з big data є певним стандартом в сфері, тому що, компаніям складно самотужки знайти достатньо обчислювальних ресурсів для обробки великого обсягу інформації, але у цього підходу також існують і недоліки [14]:

– безпека даних. Однією з найбільших загроз є питання безпеки даних. Для обробки великих обсягів чутливих даних в хмарі потрібно вдосконалені заходи безпеки, і не всі постачальники гарантують однаковий рівень захисту;

– приватність даних. Використання хмарних послуг може підвищити питання щодо приватності даних. Користувачі можуть відчувати себе некомфортно, знаючи, що їх дані зберігаються на серверах третіх сторін;

– залежність від постачальника. Компанія стає залежною від постачальника хмарних послуг. Якщо виникають проблеми з послугою або постачальником, це може призвести до перебоїв у роботі. Також, процес зміни постачальника може бути довготривалим та дороговартісним, наприклад деякі хмарні провайдери мають унікальні технології яким складно знайти альтернативу поза екосистемою цього провайдера, використання технологій може призвести до великих витрат в разі міграції до іншого провайдера, тому що буде потрібно змінювати архітектуру застосунку відповідно до нових технологій;

– затримка. Передача даних через мережу може спричинити затримки, особливо якщо великі обсяги даних повинні подолати велику відстань. Це

може бути проблемою для завдань, які вимагають низької затримки, таких як обробка даних в реальному часі;

– витрати на передачу даних. Передача великих обсягів даних з хмари може призвести до додаткових витрат на передачу даних через мережу, що може бути значущими для деяких організацій;

– можливість обмежень щодо місця розташування даних. Деякі країни та організації можуть мати обмеження щодо того, де можуть бути збережені та оброблені їхні дані, що може створювати юридичні та регуляторні складнощі;

– втрата контролю. Використання хмарних ресурсів може призвести до втрати контролю над апаратною та програмною інфраструктурою, що може бути некорисним для специфічних завдань або організацій;

– витрати на обслуговування. Хоча користування хмарними послугами знижує початкові витрати на обладнання, витрати на їх обслуговування та місячні платежі також можуть збільшуватися з часом.

Виходячи з аналізу, можна зазначити, що для забезпечення постійної конкурентоспроможності бізнесу необхідно використовувати хмарні технології для роботи з big data.

1.4 Розробка big data pipelines за допомогою хмарних технологій

Процес обробки big data виконується в рамках конвеєра даних (data pipeline). Data pipeline – це концепція та інфраструктура для автоматизованого збору, обробки, перетворення та передачі даних з одного джерела до іншого або з одного етапу обробки до іншого. Його ціль – забезпечити плавний потік даних від джерела до місця їх використання або зберігання.

Необроблені дані збираються з різних джерел та подаються до сховища даних, такого як Data Lake або Data Warehouse, для подальшого

аналізу. Перед тим, як дані надійдуть до сховища, вони зазвичай піддаються певній обробці. Це може включати в себе перетворення даних, такі як фільтрація, маскування та агрегація, для забезпечення їх інтеграції та стандартизації. Це особливо важливо, коли дані призначені для реляційної бази даних, оскільки вона вимагає певної структури і відповідності типів даних [5].

Конвеєри даних функціонують як «трубопроводи» (pipes) для наукових досліджень даних або аналітики для бізнесу. Дані можуть надходити з різних джерел, таких як API, SQL і NoSQL бази даних, або файли, але вони часто не готові для негайного використання. Задача обробки даних зазвичай лежить на плечах вчених даних або інженерів даних, які перетворюють дані відповідно до бізнес-потреб. Тип обробки даних, необхідний для конвеєра, зазвичай визначається поєднанням наукового аналізу даних та бізнес-вимог.

Після того, як дані відфільтровані, об’єднані та узагальнені, вони можуть бути збережені та використані. Організовані конвеєри даних надають підґрунтя для різноманітних проектів з аналізу даних, включаючи в себе пошуковий аналіз даних, візуалізацію даних та завдання машинного навчання.

Загалом, data pipeline складається із процесів та компонентів які зазначені на рисунку 1.2 [10, **Error! Reference source not found.**]. Це послідовність кроків та компонетів, які допомагають збирати, обробляти та відображати дані:

– джерела даних (data source). Це може бути зовнішні джерела, такі як датчики, API, вебсайти, бази даних, або внутрішні джерела, такі як локальні системи запису. Збирання може включати в себе регулярні запити для отримання нових даних або періодичні завантаження;

– поглинання (ingestion). На цьому етапі дані переносяться з джерела до сховища даних. Дані можуть бути інтегровані з джерелом у реальному часі (streaming) або періодично збиратися та імпортуватися в систему (batch);

– зберігання даних (data storage). Часто необроблені дані зберігаються в dale lake, потім вони очищаються, дублікати та аномалії видаляються та трансформуються відповідно до схеми. Нарешті, ці готові до використання дані зберігаються в data warehouse;

– обчислення (computation**)**. На цьому етапі дані піддаються аналізу, обчисленням і витягненню цінної інформації. Це може включати в себе створення звітів, агрегацію даних, виконання аналізу рядів часу, машинне навчання та інші обчислювальні операції;

– презентація (presentation). На останньому етапі дані відображаються для користувача або інших систем. Це може включати в себе застосування business intelligence платформ, створення звітів, візуалізацію даних, створення інформаційних панелей, вебсайтів або API для доступу до даних. Представлення даних допомагає користувачам отримувати доступ до аналітичної інформації та приймати рішення на основі оброблених даних.

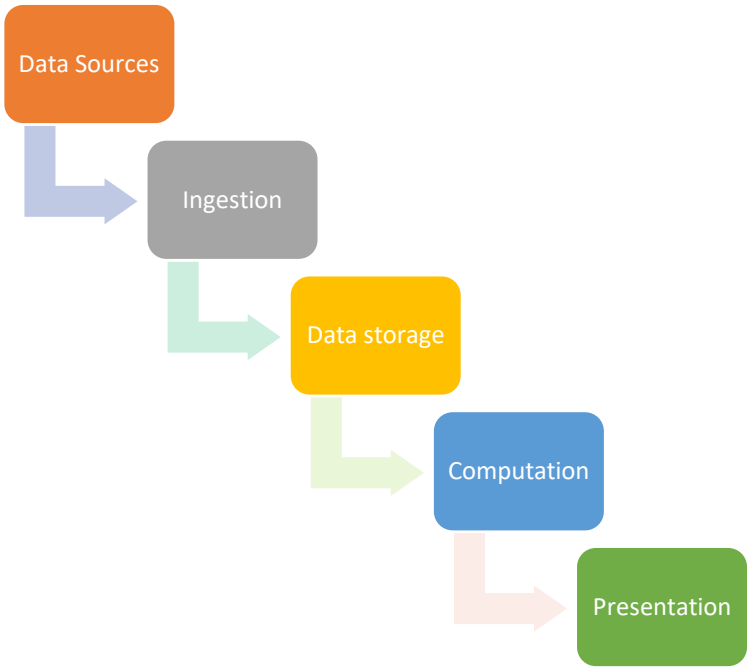

Рисунок 1.2 – Загальна модель конвеєру даних

Етапи конвеєру даних можуть варіюватися залежно від конкретного використання і потреби в аналізі даних, але загальна ідея полягає в тому, щоб

забезпечити послідовний та ефективний потік даних від джерела до результату.

Основними типами конвеєрів даних є batch (пакетна) та streaming (потокова) обробка даних. Кожен з цих типів має свої особливості та використання [17].

Пакетна обробка передбачає обробку даних у великих блоках. Дані збираються, зберігаються та обробляються у пакетах на основі певного розкладу, часто щодня, щотижня або за іншими інтервалами. Цей підхід добре підходить для обробки великих обсягів даних, таких як створення звітів за певний період часу.

Потокова обробка передбачає обробку даних у реальному часі, по мірі їхнього надходження. Дані обробляються без затримок, і результати аналізу можуть бути доступні негайно. Цей підхід добре підходить для випадків, коли необхідно відстежувати та реагувати на події в реальному часі, наприклад, аналіз потоку транзакцій, моніторинг сенсорних даних або виявлення аномалій. Для потокової обробки використовуються спеціалізовані інструменти та технології, такі як Apache Kafka, Apache Flink, Apache Spark Streaming тощо.

Основні відмінності між пакетною та потоковою обробкою:

– час обробки. В пакетній обробці дані акумулюються та обробляються зі затримкою, тоді як у потоковій обробці обробка відбувається в режимі реального часу;

– застосування. Пакетна обробка підходить для аналізу даних за певними інтервалами, тоді як потокова обробка використовується для реального відстеження подій і потоку даних;

– складність. Потокова обробка зазвичай потребує більшої складності в розробці та обслуговуванні через реальний час обробки.

Обираючи між цими двома підходами, потрібно враховувати конкретні потреби проєкту.

Хмарні технології можуть бути використані при реалізації конвеєра даних на кожному із його етапів. На етапі збору, можна використовувати хмарні сервіси для збору даних наприклад, AWS API Gateway для розгортання REST API, щоб приймати дані зі зовнішніх джерел, або сервіси для розгортання відповідних застосунків для збору даних. Для поглинання можна застосовувати такі хмарні інструменти, як AWS Kinesis або Azure Stream Analytics, які можуть бути використані для прийому та обробки даних в реальному часі. AWS EMR, AWS Glue можуть бути використані при пакетній обробці для ETL (Extract, Transform, Load) завдань. Коли дані завантажені до Date Lake, яким може виступати AWS S3, може виникнути потреба в очищені цих даних перед завантаженням до Data Warehouse, для розгортання якого можна використовувати такі сервіси як AWS Redshift, Snowflake, BigQuery, для цього також можна скористатися сервісами для пакетної обробки даних.

На етапі обчислення операції можуть бути проведені за допомогою AWS Lambda Functions або звичайних застосунків які розгорнуті на віртуальних машинах (AWS EC2), також можуть бути виконані завдання машинного навчання за допомогою таких сервісів як AWS SageMaker, Azure ML.

На етапі презентації можуть використані такі business intelligence рішення, як PoweBI, Amazon QuickSight, Looker [18–20].

Наприклад, архітектура big data pipeline для збору та аналізу статистки користування певним застосунком з використанням AWS сервісів може бути побудована як зазначено на рисунку 1.3.

В даній архітектурі для збору даних використовується AWS API Gateway, для поглинання даних Kinesis Data Firehose, для зберігання необроблених даних в роді Date Lake використовується AWS S3, для обробки даних, очищення, фільтрації та їх подальшого завантаження до Data Warehouse використовуються AWS EMR, який може бути використаний для запуска Apache Spark застосунків для пакетної обробки даних. Amazon

Redshift виступає в ролі Data Warehouse, для зберігання структурованих даних, а для візуалізації даних використовується Amazon QuickSight.

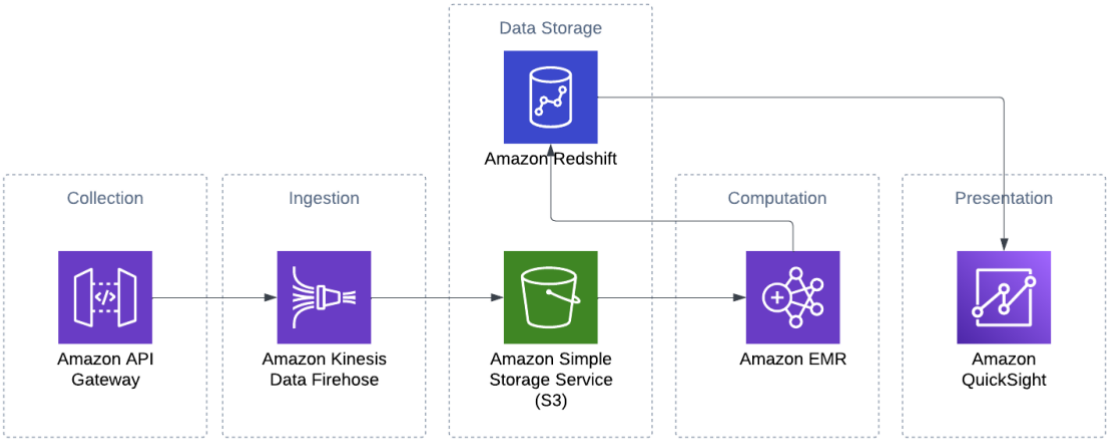

Рисунок 1.3 – Архітектура реалізації data pipeline з використання AWS сервісів

Такий метод збору інформації може бути застосований в сферах де точність даних про активність користувачів дуже важлива. Наприклад, в рамках платформи дистанційного навчання запропонований підхід допомагає визначити, скільки часу студенти витратили на проходження курсу, що для деяких дисциплін є показником того, чи надавати студентові сертифікат про успішне проходження.

1.5 Аналіз провайдерів хмарних технологій

Компанії шукають оптимальні рішення для обробки та аналізу великих обсягів даних та намагаються вибрати найкращого провайдера хмарних послуг. Серед провайдерів, які вирізняються на цьому ринку, особливе місце займають Amazon Web Services (AWS), Microsoft Azure та Google Cloud Platform (GCP). Ця трійка провайдерів має найбільшу частку ринку (рис. 1.4).

AWS вважається однією з найбільш універсальних хмарних платформ. Вона пропонує широкий спектр послуг, починаючи від створення інфраструктури і закінчуючи розробкою програмного забезпечення. AWS – це платформа, яка першою представила хмарні технології широкому загалу, і визначила модель ціноутворення на основі використаних послуг. AWS надає великий рівень контролю та гнучкості, що особливо важливо для великих обсягів даних [15].

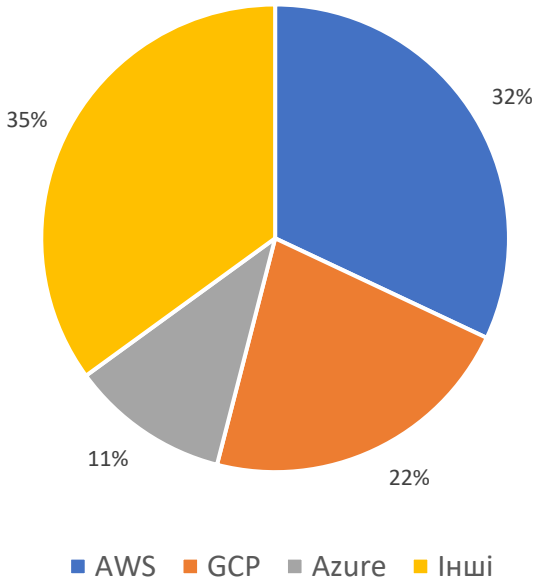

Рисунок 1.4 – Розподілення ринку між основними хмарними провайдерами

Перевагами AWS є:

– широкий спектр послуг. AWS має багатий портфель послуг і інструментів, включаючи обчислення, сховища, бази даних, аналітику, машинне навчання, послуги розробки хмарних застосунків. Він охоплює потреби компаній з різними вимогами та розмірами;

– масштабування за потребою та гнучкість. Залежно від навантаження компанії може масштабувати ресурси вгору або вниз. Розглядаючи поточний попит, компанії можуть налаштовувати використання сховищ, обчислювальну потужність та мережеві ресурси;

– глобальне покриття. AWS має обширну інфраструктуру, охоплюючи багато регіонів у всьому світі. Це забезпечує доступ з низькою затримкою, що гарантує продуктивність та плавний досвід користувача;

– безпека та відповідність. Постачальник пропонує широкий спектр служб забезпечення безпеки та сертифікацій відповідності для послуг розробки вебзастосунків та більше. Це забезпечує безпеку та безперервну роботу застосунків і даних [21, 22].

Що стосується недоліків то можна зазначити наступні.

– складність. Різноманітність пропозицій може зробити вибір правильних послуг важким для тих, хто новий у хмарі. Для розуміння та вибору послуг AWS може знадобитися додаткова глибока підготовка та експертиза;

– залежність від сервісів. Якщо користувач великою мірою покладається на послуги AWS, перехід на іншого провайдера може бути важким в разі потреби через тенденції ринку обчислювання в хмарі;

– деякі сервіси мають ліміт використання, як стосуються кількості запитів, або часу виконання цього запиту, більшість компаній ніколи не досягають цих лімітів, але якщо потрібно збільшити ліміти то можна зробити запит до AWS підтримки.

Іншим конкурентом AWS є Azure. Це хмарна платформа від Microsoft з широким спектром послуг. Azure дозволяє користувачам розробляти додатки та запускати існуючі рішення в хмарі. Вона пропонує інструменти, що підходять для різних галузей та використовує відкриті технології. Microsoft надає потужну екосистему у сфері обробки даних, що робить Azure привабливим вибором для big data [19].

Були визначено наступні переваги Azure:

– Azure надає різноманітні послуги хмарних технологій та рішення, включаючи віртуальні машини, бази даних, штучний інтелект та машинне навчання, аналітику та багато іншого. Це дозволяє задовольнити різноманітні

потреби бізнесу та вибирати оптимальні інструменти для вирішення завдань з обробки великих обсягів даних;

– інтеграція з екосистемою Microsoft. Azure добре інтегрується з іншими продуктами та сервісами Microsoft, такими як Windows Server, Active Directory та Office 365. Це спрощує управління та дозволяє безперервну співпрацю всередині екосистеми Microsoft;

– глобальна присутність. Azure має значне глобальне розміщення з розподіленими дата-центрами у багатьох регіонах всього світу. Це дозволяє організаціям розгортати свої додатки ближче до кінцевих користувачів, забезпечуючи низьку затримку та покращену продуктивність [16].

Виділено і певні недоліки:

– складність. Як і інші хмарні платформи, Azure може бути складним, особливо для початківців чи тих, хто не має досвіду з хмарами. Розуміння та ефективне управління ресурсами Azure може вимагати спеціалізованої підготовки та експертизи;

– управління витратами. Хоча Azure надає інструменти для оптимізації витрат, розуміння та управління витратами в динамічному хмарному середовищі може бути складним. Необхідно постійно моніторити використання ресурсів та впроваджувати відповідні практики управління витратами, щоб уникнути несподіваних витрат;

– документація та підтримка. Хоча Azure пропонує обширну документацію та ресурси, деякі користувачі можуть знайти документацію складною або перевантаженою. Крім того, рівень підтримки може варіюватися залежно від обраного плану підтримки, і користувачам може знадобитися покладатися на форуми спільнот.

GCP є набором хмарних інструментів, що працюють в одному середовищі з Google Search і YouTube. GCP пропонує послуги для обчислення, зберігання та розробки додатків [20].

Google Cloud відзначається своєю гнучкістю та потужними інструментами для обробки великих обсягів даних та машинного навчання.

До переваг можна віднести:

– масштабованість. Google Cloud надає інфраструктуру, яка легко масштабується в залежності від потреб. Користувачі можуть ефективно змінювати ресурси, враховуючи зміну попиту;

– глобальна інфраструктура. Платформа пропонує широку глобальну мережу дата-центрів, що забезпечує низьку затримку та безперебійність у різних регіонах світу;

– обробка великих обсягів даних та машинне навчання: Google Cloud надає потужні інструменти та послуги для обробки великих обсягів даних та машинного навчання, що дозволяє отримувати розширені аналітичні висновки та інсайти.

Недоліком GCP є залежність від власних технологій Google і обмежений вибір мов програмування відлякують деяких користувачів. Перехід від GCP до іншого постачальника є надто складним.

AWS, Azure та GCP активно розширюють свою глобальну присутність для забезпечення стабільності своїх послуг. Наприклад, AWS має 102 зони доступності в 32 географічних регіонах світу, і планує розширити це число. Azure працює в 59 регіонах і має 113 зон доступності в дії. Google Cloud планує встановити 133 зони доступності до кінця 2024 року. Крім того, ці провайдери мають значні ресурси та досвід у галузі обробки великих обсягів даних [22].

Обираючи провайдера хмарних послуг для big data, організації повинні враховувати свої конкретні потреби, рівень експертизи свого персоналу і бюджет. Кожен з цих провайдерів має можливості для розв’язання завдань big data, але варто ретельно вивчити їхні можливості та вартість, щоб знайти оптимальне рішення для свого бізнесу.

AWS відзначається високим рівнем безпеки і надійності своїх послуг, що є критичним для збереження цілісності та конфіденційності даних.

Також важливо додати, що AWS надає широкий набір безкоштовних ресурсів для навчання, таких як безкоштовні курси та матеріали. Також,

сертифікаційна програма AWS Certification дозволяє отримати практичні навички та підтвердити їх сертифікатами.

1.6 Постановка задачі дослідження

У сучасному світі обробка та аналіз великих обсягів даних (big data) стає все важливішим завданням у багатьох галузях. Хмарні технології стали незамінним інструментом для розв’язання цих завдань, оскільки вони надають масштабовану та високопродуктивну інфраструктуру для обробки цих даних. У атестаційній роботі магістра вивчаються основні характеристики хмарних сервісів та їх вплив на вирішення конкретних бізнес задач.

Об’єктом дослідження є хмарні технології у контексті обробки та агрегації даних для побудови звітів.

Метою дослідження є вивчення та аналіз хмарних технологій AWS у контексті їх застосування для обробки та аналізу великих обсягів даних у системі для побудови звітів про успішність студентів.

Для досягнення поставленої мети потрібно вирішити наступні завдання:

– дослідити особливості хмарних технологій та їх роль у роботі з big data;

– розглянути технології AWS для обробки великих даних;

– розробити конвеєр даних, який дає можливість автоматизовано збирати дані про оцінки студентів;

– розробити та дослідити конвеєр даних для побудови звітів.

2 ЗАСТОСУВАННЯ ХМАРНИХ СЕРВІСІВ AWS ДЛЯ ЗАДАЧ BIG DATA

2.1 Методи збору даних за допомогою сервісів AWS

Amazon Web Services (AWS) пропонує великий і повністю інтегрований набір рішень для хмарних обчислень, розроблених, щоб допомагати користувачам у розробці, захисті та розгортанні програм для великих даних. З AWS немає необхідності купувати апаратне забезпечення або керувати та масштабувати інфраструктуру, що дозволяє спрямувати свої ресурси саме на реалізацію програмного забезпечення. Завдяки постійному додаванню нових можливостей і функцій користувач може постійно використовувати найновіші технології, не зобов'язуючись робити довгострокові інвестиції [15].

У традиційній інфраструктурі для критично важливих додатків розробники часто стикаються з надмірною ініціалізацією, щоб впоратися з потенційними стрибками даних через збільшення бізнес-потреб. Навпаки, AWS дозволяє швидко надавати потужність і обчислювальні ресурси. Це означає, що програми для великих даних можуть розширюватися або звужуватися відповідно до потреб, забезпечуючи оптимальну ефективність системи.

Хмарні сервіси AWS дозволяють розробити всі етапи конвеєру даних, розглянуті в попередньому розділі, із мінімальними зусиллями зі сторони розробників. Розглянемо методи збору даних за допомогою сервісів AWS.

Будь-який конвеєр даних починається із джерела інформації, ними можуть виступати застосунки, сенсори які генерують дані у відповідь на будь-яку зміну стану, наприклад коли студент відправляє завдання на навчальній платформі то ця подія є джерелом інформації про те яке завдання було відправлено, та яку оцінку було отримано за нього. Надалі такі дані потрібно зібрати та завантажити до сховища даних для її подальшого

опрацювання, наприклад, для побудови звіту про успішність студентів щодо проходження курсу.

AWS надає декілька сервісів для виконання цього завдання, яке може бути вирішено за допомогою потокової або пакетної обробки даних. Дані сервіси допомагають збирати інформацію за допомогою потокового підходу:

– Amazon Kinesis;

– Amazon Managed Streaming for Apache Kafka (Amazon MSK).

Amazon Kinesis – це платформа AWS, призначена для керування потоковими даними, яка пропонує спрощене завантаження та аналіз даних. Це також змогу розробляти програми потокового передавання даних відповідно до конкретних вимог. Kinesis дозволяє завантажувати джерела даних у режимі реального часу, зокрема журнали програм, потоки кліків вебсайтів, телеметричні дані Інтернету речей (IoT) тощо у бази даних, озера даних, сховища даних або створювати власні програми на основі цих даних. Ключова перевага сервісу полягає в тому, що він дає змогу обробляти й аналізувати дані, що надходять, в режимі реального часу, замість того, щоб накопичувати всі дані перед обробкою.

На даний момент платформа Kinesis складається з наступних компонентів для збору даних [18]:

– Amazon Kinesis Data Streams – цей сервіс полегшує створення спеціальних програм для обробки та аналізу потокових даних;

– Amazon Kinesis Video Streams – дає змогу розробляти спеціальні програми для обробки та аналізу потокових відеоданих;

– Amazon Kinesis Data Firehose – дозволяє передавати потокові дані в реальному часі в різні місця призначення AWS, як Amazon S3, Amazon Redshift, OpenSearch Service і Splunk.

Amazon Kinesis Data Streams (KDS) дозволяє збирати дані із джерел даних і виконувати безперервну обробку. Ця обробка може передбачати перетворення даних перед пересиланням їх до іншого сховища даних, аналітику в реальному часі або консолідацію кількох потоків у більш складні

потоки для подальшої обробки. Нижче наведено типові випадки використання Kinesis Data Streams в аналітичних завданнях:

– аналітика даних у режимі реального часу. KDS полегшує аналіз потокових даних у реальному часі, наприклад аналіз даних потоку кліків на вебсайті та аналіз залучення клієнтів;

– приймання та обробка даних. За допомогою потоків даних Kinesis продюсери даних можуть безпосередньо передавати дані в потік Amazon Kinesis. Наприклад, користувач може спрямувати системні журнали та журнали додатків до Kinesis Data Streams і отримати доступ до потоку для обробки протягом кількох секунд;

– метрики та звітність у реальному часі. Отримані дані в потоках даних Kinesis можна використовувати для отримання метрик і генерації ключових показників ефективності (KPI) для звітів у реальному часі. Це дозволяє програмі обробки даних працювати з даними, які постійно надходять, усуваючи необхідність чекати надходження пакетів даних.

Хоча Kinesis Data Streams може вирішувати численні проблеми з потоковими даними, найпоширенішим випадком використання передбачає виконання агрегації даних у реальному часі та подальшу передачу агрегованих даних до сховища даних або кластера розподіленої обробки даних. Джерело даних, наприклад, застосунок для відображення тестових завдань для студентів, відправляє дані до KDS за допомого Kinesis Producer Library (KPL). Далі отримувач даних повинен розробити застосунок який буде витягувати дані із KDS та опрацьовувати їх, це може включати в себе консолідацію даних, наприклад збір даних про оцінки конкретного студента, або збагачення цих даних якоюсь додатковою інформацією перед її подальшим відправленням до кінцевої точки. Такий застосунок може бути розроблений на основі таких AWS сервісів як Amazon EMR (шляхом розгортання Apache Spark програм), Amazon EC2 (з використання Kinesis Client Library), AWS Lambda, Amazon Kinesis Data Analytics. Після того як

дані були опрацьовані результат зберігається до сховища даних, який потім може бути проаналізований за допомогою Business Intelligence застосунків.

Також важливою особливістю KDS є те, що кілька додатків можуть незалежно споживати дані з потоку, забезпечуючи одночасну діяльність, наприклад архівування та обробку. Наприклад, дві програми можуть читати дані з одного потоку [18].

Схема збору даних за допомогою KDS зазначена на рисунку 2.1.

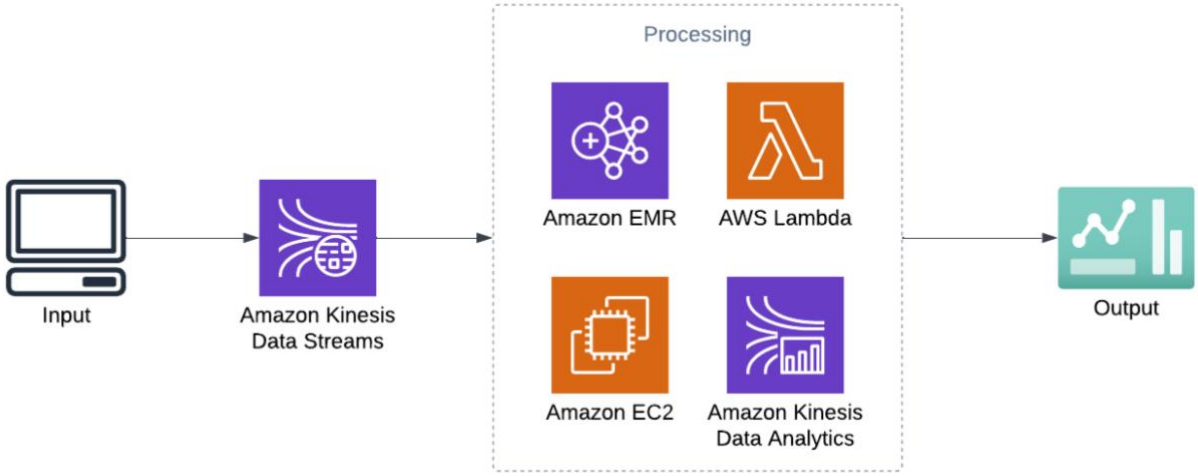

Рисунок 2.1 – Модель збору даних за допомогою Kinesis Data Streams

Amazon Kinesis Video Streams (KVS) спрощує безпечну передачу відеоданих із підключених пристроїв до AWS, сприяючи різноманітним процесам, таким як аналітика, машинне навчання (ML). KVS автоматично налаштовує та масштабує необхідну інфраструктуру для прийому потокових відеоданих із безлічі пристроїв. Також даний сервіс дає змогу відтворювати відео в прямому ефірі та на вимогу, швидко розробляти програми, які використовують комп’ютерний зір, через інтеграцією з Amazon Rekognition Video та бібліотеками, сумісними з популярними фреймворками ML, такими як Apache MxNet, TensorFlow і OpenCV (рис. 2.2).

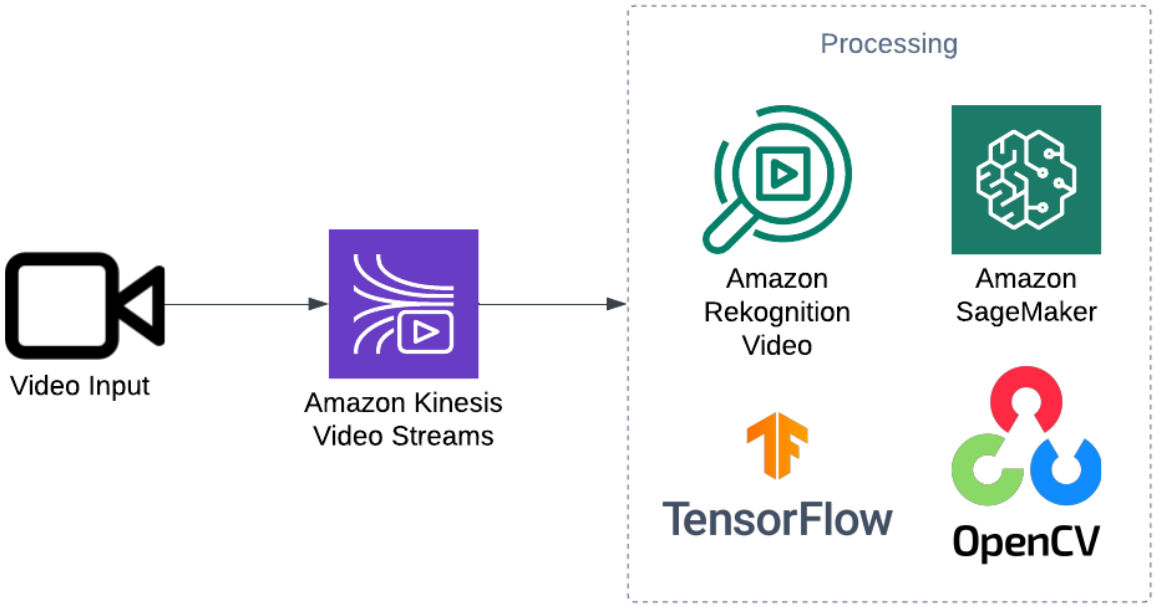

Рисунок 2.2 – Модель обробки відеоданих за допомогою Kinesis Video Streams

За допомогою даного сервісу та його інтеграції з бібліотеками машинного навчання можна вирішувати завдання аналізу відеоданих у реальному часі та реагування на зміну стану, наприклад, його можна застосовувати для аналізу завантаженості трафіку на дорогах, запобігати злочинам шляхом розсилання екстрених оповіщень.

Останнім сервісом для збору даних в рамках платформи Kinesis, який буде розглянуто, є Amazon Kinesis Data Firehose (KDF). Цей сервіс дає можливість ефективно доставляти потокові дані у режимі реального часу в різні місця призначення, такі Amazon S3, Amazon Redshift, Amazon OpenSearch Service, Amazon OpenSearch Serverless, Splunk і спеціальні кінцеві точки HTTP.

Ключовою перевагою використання KDF є те, що він дозволяє уникнути складнощів розробки програм і управління ресурсами. Замість цього користувачу необхідно налаштовати продюсерів даних для передачі даних до KDF, який у свою чергу автоматично завантажує дані до пункту призначення, варто зазначити, що на відміну від KDS дані не звантажуються

в реальному часі, натомість є мінімальний період буферизації в одну хвилину.

Загалом, цей сервіс надає змогу з легкістю розробити ETL конвеєр даних, тобто отримати, трансформувати, та завантажити дані то пункту призначення (рис. 2.3) [23].

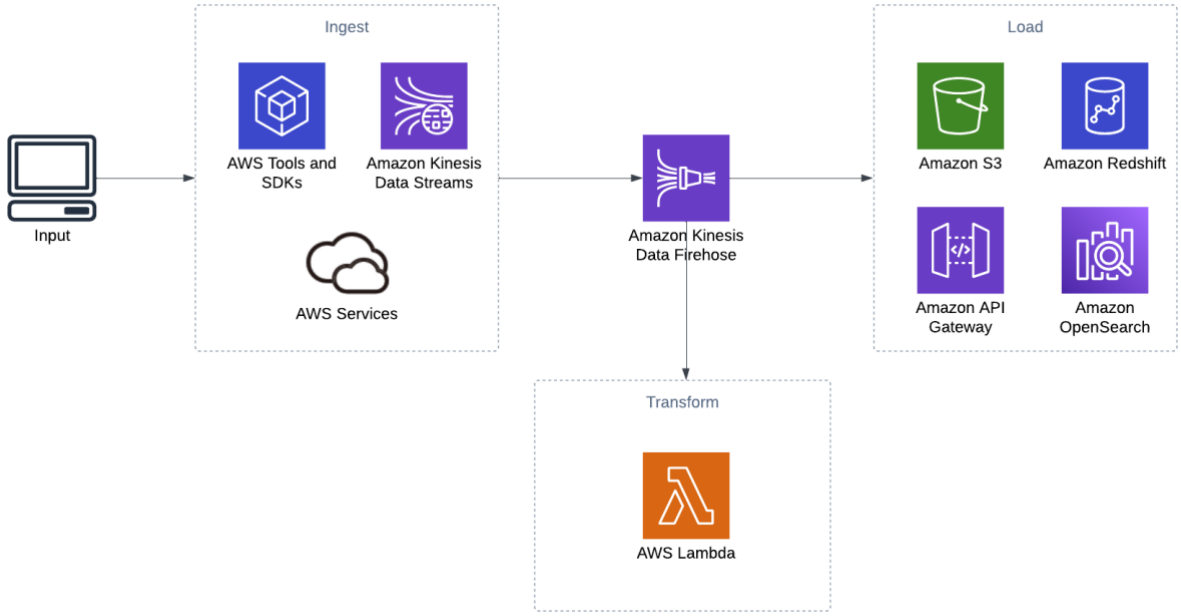

Рисунок 2.3 – Схема збору даних за допомогою Kinesis Data Firehose

Дані потрапляють до KDF із різних AWS сервісів, наприклад KDS може виступати джерелом, далі вони можуть бути трансформовані опціонально за допомогою AWS Lambda. Вхідні дані збираються деякий проміжок часу, від 1 до 15 хвилин, та в пакетному вигляді відправляються до пункту призначення, яким можуть бути Amazon S3, Amazon Redshift та інші сервіси AWS.

Amazon Kinesis є власним рішенням AWS для обробки потових даних, окрім цього AWS підтримує розгортання Apache Kafka для виконання цього завдання за допомогою Amazon MSK. Apache Kafka, платформа з відкритим вихідним кодом для створення конвеєрів потокових даних. Використовуючи

Amazon MSK, можна використовувати API Apache Kafka для заповнення озер даних, та розробки програм машинного навчання та аналітики.

Amazon MSK спрощує процес створення та експлуатації програм які використовують Apache Kafka, усуваючи потребу в спеціальних навичках керування інфраструктурою Apache Kafka. (рис 2.4) [18].

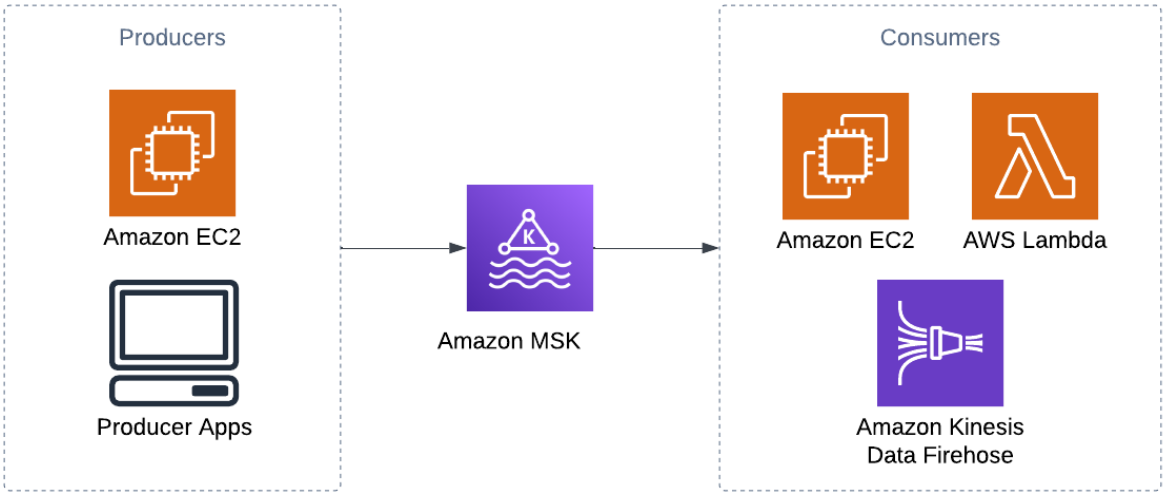

Рисунок 2.4 – Модель збору даних з використанням Amazon MSK

Загалом модель збору даних за допомогою Amazon MSK подібна до KDS, адже ці сервіси вирішують однакові завдання, що стосуються потокової обробки даних.

Окрім потокового збору даних можна також застосовувати пакетний підхід, тобто коли дані надходять до пунктів призначення не в реальному часі, а за певним розкладом, наприклад, це може бути завантаження інформації із бази даних застосунку до сховища даних. Для виконання цього завдання AWS надає такі сервіси як Amazon EMR та AWS Glue, які будуть розглянуті у розділі про обробку даних, так як збір даних та їх обробка пов’язані між собою в контексті пакетних конвеєрів даних.

2.2 Зберігання великих даних за допомогою сервісів AWS

Для зберігання великого обсягу даних AWS надає такі сервіси як Amazon S3, який дозволяє будувати озера даних, тобто зберігати неструктуровані дані, та Amazon Redshift, який виступає в ролі сховища структурованих даних (Data Warehouse). Розглянемо більш детально дані технології [24].

Amazon S3 – це сервіс зберігання інформації, де дані зберігаються у формі об’єктів, що містяться в «бакетах». Кожен об’єкт по суті є файлом, разом із пов’язаними метаданими, що описують файл. «Бакети» – це сховища цих об’єктів.

Щоб зберігати дані в Amazon S3, потрібно створити бакет і вказати унікальне ім’я для нього. Потім користувач буде мати змогу завантажувати свої дані в цей бакет як окремі об’єкти. Кожен об’єкт ідентифікується за допомогою ключа, який служить ідентифікатором для об’єкта в межах даного бакета.

Процес роботи з Amazon S3 полягає в наступному (рис 2.5) [18]:

– розробники big data застосунку налаштовують збирання даних які потрібні для нього в Amazon S3 шляхом використання технологій які були зазначені в попередньому підрозділі, це можуть бути будь-які дані, JSON, XML файли, відео, зображення;

– по мірі того, як ці дані надходять до системи, вони зберігаються в «сирому» форматі в Amazon S3, тобто в неструктурованому виді;

– далі отримані опрацьовуються big data застосунками для того, щоб структурувати дані та отримати із них корисну інформацію. Це може включати широкий набір операцій та етапів, наприклад, дані можуть бути трансформовані відповідно до структури, необхідної для сховища даних, з їх подальшим завантаженням для історичного зберігання та проведення Business Intelligence аналізу. Також такі дані можуть слугувати вибіркою для машинного навчання та побудови моделей.

Рисунок 2.5 – Модель зберігання даних в Amazon S3

Amazon S3 має ряд переваг над іншими сервісами для зберігання даних:

– масштабованість. Amazon S3 має високу масштабованість, що дозволяє зберігати величезні обсяги даних. Незалежно від того, чи потрібно зберігати невеликий набір даних чи керувати великими озерами даних, його можна легко масштабувати відповідно до потреб;

– надійність і доступність даних. Amazon S3 забезпечує довговічність і високу доступність даних. Він копіює дані в кількох зонах доступності, зменшуючи ризик втрати даних через апаратні збої;

– безпека. Amazon S3 пропонує надійні засоби контролю доступу, шифрування даних у стані спокою та під час передачі та інтегрується з AWS Identity and Access Management (IAM) для детального керування доступом, забезпечуючи конфіденційність і цілісність ваших даних;

– універсальність. Amazon S3 підтримує широкий спектр варіантів використання. Він використовується для архівування даних, резервного копіювання та відновлення, розміщення статичних вебсайтів, обслуговування вмісту для веб і мобільних додатків, озер даних і аналітики великих даних;

– економічна ефективність. Amazon S3 працює за моделлю оплати за використання. Користувач платє лише за сховище та передачу даних, які

фактично використовуються, що робить це економічно ефективним рішенням для компаній будь-якого розміру.

Варто зазначити, що Amazon S3 не є заміною баз даних, це сховище файлів, які містять неструктуровані дані, тому Amazon S3 не варто використовувати в випадку якщо застосунку потрібний швидкий доступ до певної вибірки даних, Amazon S3 має більшу затримку для виконання цього завдання, аніж традиційні бази даних [18].

Amazon Redshift, на відміну від Amazon S3, призначений для зберігання структурованих даних. Він дозволяє зберігати великі обсяги даних і виконання складних запитів на високій швидкості. Ключовими особливостями даного сервісу є [24]:

– масштабованість. Amazon Redshift може легко масштабувати дані від кількох сотень гігабайт до петабайтів, а його продуктивність змінюється лінійно залежно від розміру кластера. Ця гнучкість дозволяє підприємствам розвиватися та адаптувати свої потреби в сховищах даних по мірі зростання обсягу даних;

– продуктивність. Amazon Redshift оптимізовано для високопродуктивних запитів. Він використовує паралельну обробку та стиснення даних для швидкого аналізу та отримання даних. Він особливо добре підходить для складної аналітики та звітності з інтенсивним об’ємом даних;

– інтеграція з BI інструментами: Amazon Redshift легко інтегрується з популярними інструментами такими як Tableau, Qlik і Amazon QuickSight, що дозволяє користувачам створювати переконливі візуалізації та звіти.

На рівні прийому даних різні типи джерел постійно завантажують структуровані, напівструктуровані або неструктуровані дані на відповідний рівень зберігання. Ця область зберігання даних служить проміжною областю, яка зберігає дані в різних станах готовності до споживання. Прикладом сховища може бути відро Amazon Simple Storage Service (Amazon S3).

Процес обробки даних за допомогою Amazon Redshift показано на рисунку 2.6.

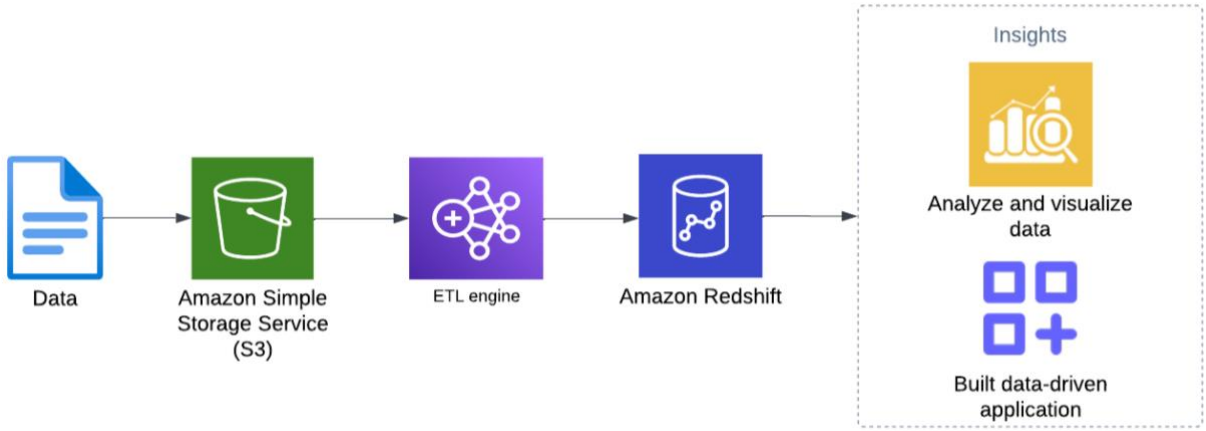

Рисунок 2.6 – Схема зберігання даних в Amazon Redshift

Дані у неструктурованому або у напівструктурованому вигляді потрапляють до озера даних, яким виступає Amazon S3, потім ETL застосунок трансформує їх та завантажує до Amazon Redshift, для їх історичного зберігання або подальшого аналізу, побудови звітів за допомогою BI інструментів, або для розробки застосунків на основі цих даних.

2.3 AWS технології для аналізу та обробки великих даних

Обробка великих даних включає в себе такі процеси, як фільтрація, перетворення, агрегація, індексація і очищення даних. Наприклад, розглядаючи платформу онлайн освіти як джерело великих даних, цілком можливо, що для виконання бізнес-завдання, щодо побудови звітів про успішність студентів, непотрібно включати в кінцевий результат оцінки незавершених завдань, то на етапі обробки можна відфільтрувати такі дані, потім трансформувати їх до певної структури яка необхідна для завантаження в сховище даних. Наступним кроком є агрегація, наприклад,

якщо необхідно розрахувати оцінку студента за курс, то потрібно загрегувати дані по кожному зданому завданню, щоб отримати звіт [9].

Для виконання цих завдань AWS надає ряд сервісів [18]:

– AWS Lambda;

– AWS Glue;

– Amazon EMR;

– Amazon Kinesis Data Analytics.

AWS Lambda – це serverless сервіс, який дозволяє виконувати код у відповідь на певні події, та автоматизувати обчислення в хмарному середовищі без необхідності керувати інфраструктурою. В контексті big data, як правило, такими подіями є надходження нових даних. AWS Lambda може бути інтегрований з понад 200 різноманітних AWS сервісів, що робить даний сервіс дуже популярним [25].

Serverless – це підхід до розробки програмного забезпечення, при якому розробники можуть писати код і не витрачати час на управління інфраструктурою, такою як сервери або віртуальні машини. Термін не означає відсутність серверів, вони все ще існують, але розробникам не потрібно про них турбуватись, завдяки цьому вони можуть спрямовувати свою увагу на написання коду та розробки функціональності програми. Також важливою перевагою цього підходу є те, що оплата відбувається за фактом використання, тобто користувач платить тільки за час коли відбуваються обчислення. Традиційний підхід з розгортанням серверів на віртуальних машинах змушує користувача платити за весь час роботи сервера, навіть якщо він не виконує корисних обчислень [8].

Наприклад, для вебзастосунку для оброблення зображень, функціонал якого включає в себе накладення фільтрів, генерацію копій різних розмірів, не потрібно підіймати традиційний сервер із використанням віртуальної машини для запуску програми, яка буде виконувати дані завдання, якщо навантаження непостійне на протязі дня, натомість для того, щоб заощадити

кошти, можна використати AWS Lambda [26-45]. Такий конвеєр обробки даних можна реалізувати наступним чином (рис. 2.7):

– користувач вебзастосунку завантажує зображення, яке хоче обробити певним чином, обравши відповідні опції, наприклад генерацію цього зображення у іншому розмірі, потім зображення зберігається в S3 бакеті для неопрацьованих файлів разом з інформацією про обрану опцію у вигляді метаданих об’єкту;

– AWS Lambda налаштована таким чином, що як тільки нове зображення було завантажене до S3 бакета для неопрацьованих файлів, то запускається лямбда функція, код якої генерує результат у вигляді нових зображень, які потім записуються до S3 бакета для опрацьованих файлів. Як тільки робота лямбди буде завершена, вебзатосунку відобразить згенеровані зображення користувачу, які він зможе завантажити.

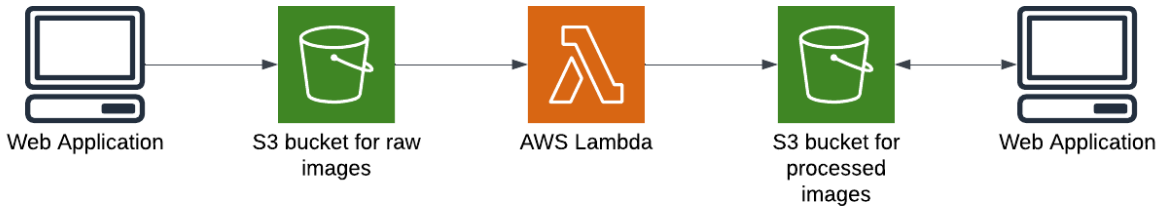

Рисунок 2.7 – Модель обробки зображень за допомогою AWS Lambda

В лістингу 2.1 наведено приклад реалізації коду лямбда функції для обробки зображень мовою програмування Java.

Лістинг 2.1 Лямбда функція для обробки зображення:

```
public class ImageHandler {
    private final S3Client s3Client  = S3Client.builder()
            .region(Region.US_EAST_1)
            .build();

    public void handleRequest(S3Event s3Event, Context context) {
```

```
        try {
            for (S3EventNotificationRecord record : s3Event.getRecords()) {
                String bucketName = record.getS3().getBucket().getName();
                String objectKey = record.getS3().getObject().getKey();

                GetObjectRequest getObjectRequest = GetObjectRequest.builder()
                        .bucket(bucketName)
                        .key(objectKey)
                        .build();

                // Fetch an image from S3
                var objectAsBytes = s3Client.getObjectAsBytes(getObjectRequest);
                byte[] image = objectAsBytes.asByteArray();

                // process the image according to selected options
                // and then save generated images to
                // the S3 bucket for generated images
            }
        } catch (Exception e) {
            // error handling
        }
    }
}
```

AWS Lambda сервіс може бути застосований також для обробки потових даних в інтеграції разом з Kinesis Data Streams, який був розглянутий в попередньому підрозділі. Нові дані які потрапляють в KDS опрацьовуються в режимі реального часу відповідною лямбда функцією, це може включати в себе очищення, збагачення та фільтрацію даних. Весь процес роботи з KDS

керується автоматично, це дозволяє лямбда функції зосереджуватися насамперед на обробці основної бізнес-логіки. Наприклад, застосунок може приймати потокові записи, що містять IP-адреси, і доповнювати їх географічною інформацією за допомогою лямбда функції, в цьому випадку відбувається процес збагачення. Іншим прикладом є опрацювання оцінок студентів в реальному часі в рамках платформи онлайн освіти (рис. 2.8) [25]:

– як тільки завдання було пройдено студентом, інформація про нього (event) відправляється до Kinesis Data Streams сервісом навчальної платформи, що відповідає за відображення завдань;

– лямбда функція запускається у відповідь на надходження нових даних та опрацьовує інформацію про конкретне завдання, що включає в себе її збагачення певними метаданими які можуть отримані із зовнішніх сервісів, наприклад це можуть бути налаштування, або більш детальна інформація про проходження певного питання;

– згенеровані дані відправляються через HTTP запит до окремого сервісу системи, що відповідальний за оброблення та зберігання оцінок в базі даних.

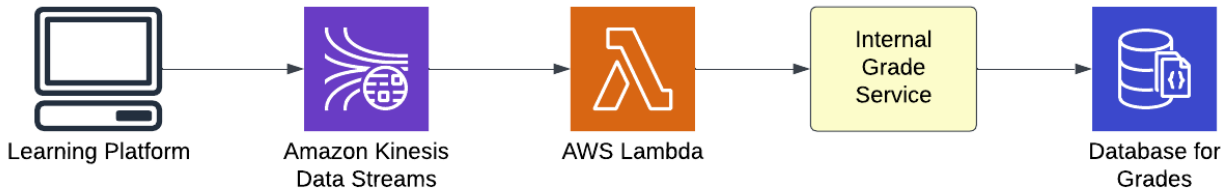

Рисунок 2.8 – Обробка оцінок студентів за допомогою AWS Lambda та Kinesis Data Streams

Також AWS Lambda надає змогу агрегувати потокові дані між собою за певний час (stateful stream processing), наприклад, збирати оцінки студента за деякий період та опрацьовувати їх певним чином перед відправкою далі, але є ліміт в 15 хвилин, тому якщо потрібно опрацьовувати більший часовий інтервал лямбда функції не підходять для виконання цього завдання, та

потрібно використовувати інші сервіси такі як Kinesis Data Analytics (KDA), Amazon EMR, AWS Glue. KDA дозволяє запускати застосунки, засновані на Apache Flink, що є потужним фреймворком для опрацювання потокових даних та може виконувати складні агрегації [18].

Amazon EMR (Elastic MapReduce) дозволяє розгортати застосунки для опрацювання великих даних, які засновані на Apache Spark шляхом запуску кластеру із декількох серверів, які використовуються для розподілених обчислень (distributed computing). В свою чергу, Apache Spark – це фреймворк для розподілених обчислювань, призначений для обробки та аналітики великих даних. Він відомий своєю швидкістю, простотою використання та гнучкістю, і його використання широко поширене для виконання завдань великих даних та аналітики даних [23].

Amazon EMR може бути використаний для поглинання даних та їх трансформації перед подальшим завантаженням, тобто для реалізації ETL процесу, що є комбінацією етапів збору та опрацювання даних. Наприклад, в рамках платформи онлайн освіти непотрібно зберігати дані про оцінки студентів кількарічної давності в базі даних застосунку. Натомість їх потрібно завантажити до сховища даних для історичного зберігання та для бізнес аналізу за допомогою BI інструментів. Такий конвеєр даних може бути реалізований наступним чином (рис. 2.9):

– дані про оцінки студентів потрапляють до бази даних застосунку;

– програма, розроблена на основі Apache Spark фреймворку з використанням Scala, Java або Python мов програмування, запускається на EMR кластері з певною періодичністю, наприклад, один раз на добу, та завантажує записи за неорацьований період із бази даних;

– записи із бази даних фільтруються та трансформується відповідно до необхідної схеми та завантажуються до сховища даних;

– завантаженні дані готові для аналітики.

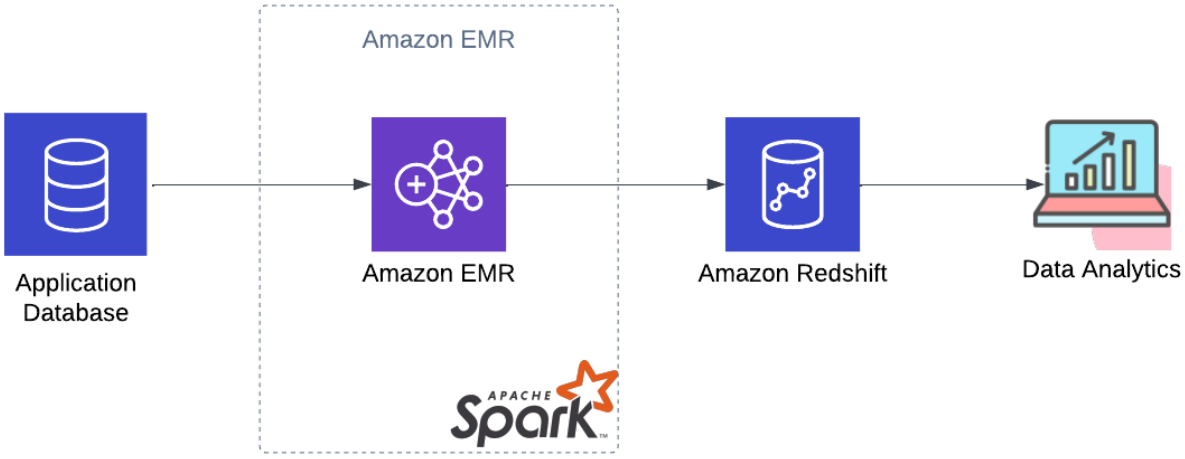

Рисунок 2.9 – Модель реалізації ETL процесу для збереження оцінок студентів в сховищі даних

Окрім трансформації, фільтрації даних, Apache Spark застосунки в комбінації з EMR кластером чудово підходять для агрегації великого обсягу даних за рахунок використання розподілених обчислень. Наприклад, процес генерації звіту про успішність студентів є процесом агрегації великої кількості даних для отримання статистичних показників.

Це може включати в себе класифікацію або кластеризацію студентів на певні групи, дані операції в Apache Spark можна здійснити за допомогою бібліотеки машинного навчання MLlib. MLlib надає широкий спектр алгоритмів та інструментів для виконання завдань машинного навчання. В лістингу 2.2 наведено приклад реалізації кластеризації методом k-середніх із використанням мови програмування Scala, який є одним із найпоширеніших алгоритмів кластеризації, що розбиває дані на попередньо визначену кількість кластерів [46].

Лістинг 2.2 Виконання кластеризації великих даних за допомогою Apache Spark:

```
val dataset = spark
  .read
  .format("libsvm")
```

```
  .load("data/mllib/sample_kmeans_data.txt")

// Train model.
val kmeans = new KMeans().setK(2).setSeed(1L)
val model = kmeans.fit(dataset)
// Make predictions
val predictions = model.transform(dataset)
// Evaluate clustering by computing Silhouette score
val evaluator = new ClusteringEvaluator()

val silhouetteScore = evaluator.evaluate(predictions)
println(s"Silhouette score = $silhouetteScore")
println("Cluster Centers: ")
model.clusterCenters.foreach(println)
```

Загалом архітектура реалізації конвеєру даних для генерації звітів за допомогою Amazon EMR має наступний вигляд:

– дані про оцінки студентів зберігаються в базі даних застосунку, а джерелом даних про активність студентів виступає S3 бакет, ці дані збираються окремими конвеєрами даних;

– Apache Spark програма запускається із заданою періодичністю відповідно до бізнес-вимог, наприклад це може відбуватися раз на тиждень якщо обсяги даних для опрацювання значні та сам процес дороговартісний;

– дані про оцінки студентів та їх активність завантажуються із відповідних джерел та апретуються Apache Spark програмою для розрахунку метрик, які необхідні для створення звіту. Це може включати в себе розрахунки оцінки кожного студента за певний курс на основі даних про кожне здане завдання та визначення обсягу часу, котрий студент витратив на проходження даного курсу;

– згенерований звіт записується до відповідного S3 бакету, який потім може бути відображений студенту або викладачу у вебзастосунку.

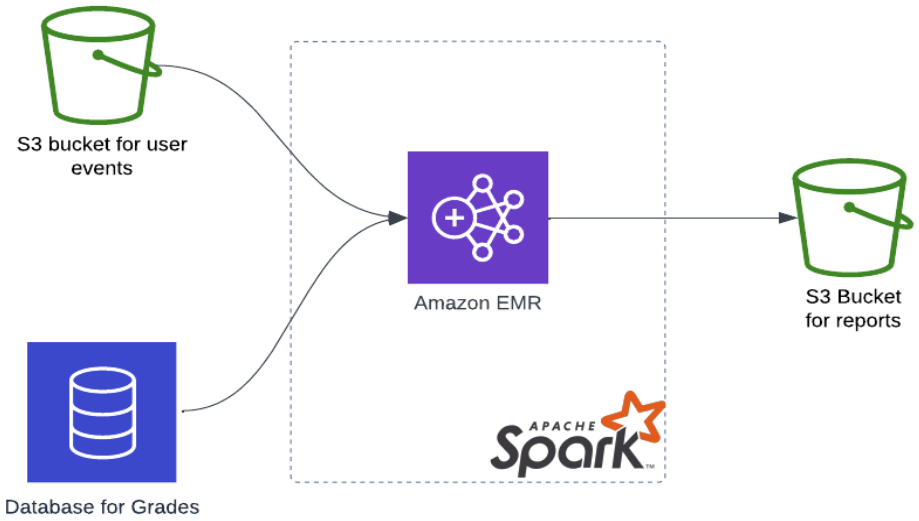

Рисунок 2.10 – Схема реалізацій конвеєру даних для генерації звітів за допомогою Amazon EMR

Що стосується AWS Glue, то це Serverless сервіс який також дозволяє розробляти ETL конвеєри даних на основі Apache Spark як і Amazon EMR. Усі процеси стосовно розгортання інфраструктури повністю керуються сервісом, що дозволяє розробникам фокусуватися тільки на розробленні бізнес-логіки програми, але це тягне за собою також і більшу ціну порівняно з EMR. Також, Amazon EMR надає прямий доступ до середовища Apache Hadoop, надаючи більший контроль та підвищену гнучкість у використанні інструментів, окрім Apache Spark. Загалом вибір між цими двома сервісами залежить від завдання яке потрібно вирішити, якщо необхідний більший контроль над ресурсами то тоді потрібно використовувати Amazon EMR [18].

3 РОЗРОБКА КОНВЕЄРІВ ДАННИХ ДЛЯ ОБРОБКИ ОЦІНОК СТУДЕНТІВ ЗА ДОПОМОГОЮ AWS СЕРВІСІВ

3.1 Моделювання та розробка конвеєру даних для поглинання оцінок студентів

Сучасні навчальні онлайн-платформи генерують великий обсяг даних, оскільки онлайн-освіта стає більш популярною з кожним роком. Для забезпечення якості цієї послуги навчальні платформи повинні швидко опрацьовувати вхідні дані, такі, наприклад, як оцінки студентів, та гарантувати їх надійне зберігання, оскільки втрачені дані про успішність студентів можуть призвести до того, що студент не зможе успішно закінчити навчальний курс. Тому важливо розробити конвеєр даних, який буде відповідати цим вимогам.

Також вдало розроблені конвеєри даних надають можливість створити персоналізовані навчальні плани для студентів. Вони можуть допомагати в ідентифікації індивідуальних потреб студентів і надавати інструменти для оптимізації навчального процесу. Це може включати аналіз успішності студентів, визначення слабких місць в навчальних програмах та розробку стратегій для покращення якості освіти.

Як результат роботи таких конвеєрів даних, викладачі та адміністрація навчальних закладів отримують детальні звіти, на основі яких можна робити висновки про успішність конкретних студентів та ефективність програми навчання загалом. Це, в свою чергу, надає можливість визначити, які методи викладання працюють краще, та сприяє вдосконаленню матеріалів певного начального курсу.

У даній кваліфікаційній роботі було розроблено декілька конвеєрів даних за допомогою AWS сервісів, які спільно працюють для виконання вищеописаного завдання.

Для розгортання хмарної інфраструктури використовувалося програмне забезпечення з відкритим вихідним кодом – Terraform.

Terraform – це інструмент для опису інфраструктури за допомогою коду (Infrastructure as Code, IaC), розроблений компанією HashiCorp. IaC – це концепція, яка передбачає використання програмованого коду для автоматизації конфігурації та управління інфраструктурою комп'ютерних систем. Замість ручного налаштування серверів та мережевих пристроїв, розробники можуть використовувати мови програмування або спеціалізовані інструменти для опису інфраструктури у вигляді коду.

Основна ідея Terraform полягає в тому, щоб дозволити розробникам описувати та управляти інфраструктурою за допомогою мови декларативної конфігурації, відомої як HashiCorp Configuration Language (HCL). Також варто зазначити, що Terraform підтримує безліч провайдерів (платформ, таких як AWS, Azure, Google Cloud, VMware, Docker, Kubernetes тощо), що дозволяє розгортати та управляти інфраструктурою на різних хмарних та локальних середовищах [47].

В першу чергу, розглянемо процес реалізації конвеєру даних для збору та зберігання оцінок студентів на навчальній платформі.

На рисунку 3.1 зображена архітектура збору оцінок студентів з навчальних платформ та їх централізованого зберігання в базі даних.

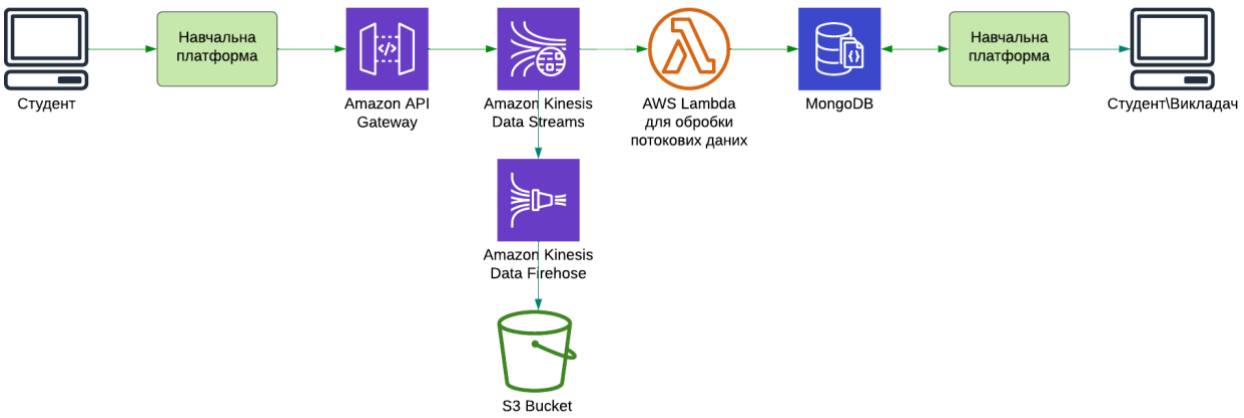

Рисунок 3.1 – Архітектура реалізацій конвеєру даних для генерації звітів за допомогою Amazon EMR

Варто відзначити, що навчальна платформа – це, зазвичай, веб-застосунок, який відповідає за надання освітніх матеріалів для студентів та відображення тестових завдань, за допомогою яких студент отримує оцінку за проходження певного курсу.

Дана система спроектована так, щоб отримувати дані у платформонезалежний спосіб, що дозволяє обслуговувати декілька незалежних навчальних платформ. Це стає корисним, якщо компанія, яка надає послуги в сфері онлайн освіти, володіє кількома різними навчальними платформами, спрямованими на надання певного типу навчального матеріалу. Наприклад, одна платформа може спеціалізуватися на курсах з вивчення іноземних мов, а інша – на математиці тощо.

Розглянемо більш детально запропоновану архітектуру та аспекти її реалізації.

Для збору оцінок студентів у цьому конвеєрі даних використовується Amazon API Gateway. Цей інструмент дозволяє навчальним платформам відправляти дані про успішність проходження завдань студентами через HTTP запити. Коли студент виконує певне завдання, навчальна платформа відправляє відповідний HTTP запит до API Gateway, який є вхідною точкою конвеєра. Приклад такого HTTP запиту представлено в лістингу 3.1.

Лістинг 3.1 Приклад HTTP запиту для відправки даних стосовно успішності студента для певного завдання:

```
curl --location --request PUT 'https://80yr5a5u99.execute-api.us-east-1.amazonaws.com/v1/streams/kinesis-stream-student-score-ingestion/record' \
--header 'Content-Type: application/json' \
--data '{
   "PartitionKey": "01ebb512-0921-4485-9d7a-0892ba0035fb",
   "Data":  {
   "userGuid": "9c47da7b-ed85-4249-830e-70cb4a86c6a9",
   "activityId": "c54f07d4-4eb0-40f9-baf6-8d497931c58a",
```

```
    "courseId": "testCourseId",
    "attemptId": "50b48059-5e96-4d14-9c10-2e731171f374",
    "score": 11,
    "maxScore": 11,
    "timeSpentInSeconds": 138,
    "submissionDate": "2023-10-05T13:27:53.745Z"
  }
}'
```

Amazon API Gateway в свою чергу інтегрується з Kinesis Data Stream. KDS дозволяє поглинати великий обсяг даних та гарантує їх надійне зберігання до 1-го року. В лістингу 3.2 наведений код на мові HQL, який дозволяє розгорнути API Gateway та KDS сервіси.

Лістинг 3.2 HQL код для розгортання API Gateway та KDS:

```
# API Gateway
module "kinesis-api-gateway" {
  source  = "dod-iac/kinesis-api-gateway/aws"
  version = "1.0.0"

  execution_role_name = "student-score-ingestion-api-role"
  name                = "student-score-ingestion-api"
  streams             = [aws_kinesis_stream.kinesis_stream.arn]
}
# Kinesis
resource "aws_kinesis_stream" "kinesis_stream" {
  name             = "kinesis-stream-student-score-ingestion"
  shard_count      = 1
  retention_period = 24
```

```
  stream_mode_details {
    stream_mode = "PROVISIONED"
  }
}
```

На рисунку 3.2 показано успішне розгортання Amazon API Gateway в консолі AWS як результат виконання HQL коду за допомогою Terraform.

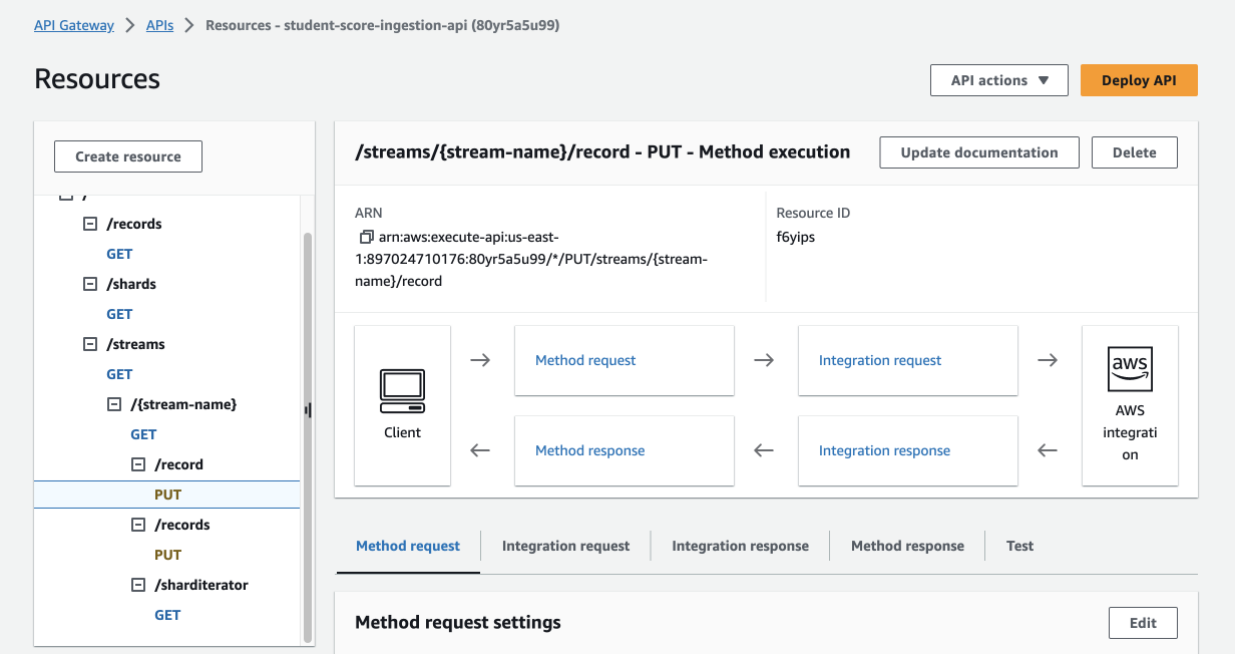

Рисунок 3.2 – Інтерфейс Amazon API Gateway в AWS консолі

В архітектурі цього конвеєру даних передбачено використання KDS (Amazon Kinesis Data Stream), який дозволяє отримувати дані з одного потоку декільком споживачам одночасно. Потік оцінок студентів обробляється двома споживачами: Amazon Kinesis Data Firehose (KDF) та AWS Lambda.

KDF відповідає за запис даних з потоку до S3 бакету для їх довготривалого зберігання. Цей підхід гарантує, що оцінки студентів не будуть втрачені в разі збою в системі, наприклад, якщо основна база даних стане недоступною. Навіть якщо дані вже були видалені з KDS через закінчення строку їх зберігання, вони залишаться доступними у S3 бакеті.

Задачею AWS Lambda в цьому конвеєрі даних є отримання оцінок з потоку даних, їх збагачення та збереження до OLTP (Online Transaction Processing) бази даних, якою виступає MongoDB.

OLTP є системою обробки транзакцій в реальному часі. Це технологічний підхід, орієнтований на ефективне та негайне виконання коротких транзакцій баз даних. OLTP використовується для операцій, які виникають в реальному часі, таких як додавання записів, оновлення і видалення даних. MongoDB, як вказано в попередньому тексті, виступає в ролі OLTP бази даних. [13].

MongoDB – це документно-орієнтована система керування базами даних (NoSQL), яка зберігає дані у форматі JSON-подібних документів з динамічними схемами. Ця база даних дозволяє зберігати та обробляти великі об'єми даних, а також швидко реагувати на зміни в структурі даних.

Основні характеристики MongoDB:

– документно-орієнтована структура. Дані в MongoDB представлені у вигляді BSON (Binary JSON) документів, які є JSON-подібними об'єктами. Кожен документ може містити різні типи даних, включаючи масиви та вкладені документи;

– динамічна схема. MongoDB використовує динамічну схему, що дозволяє документам в колекції мати різні поля. Це полегшує роботу з даними, оскільки не вимагає строго визначення схеми даних завчасно;

– масштабованість. MongoDB спроектована для горизонтального масштабування, що дозволяє легко розподіляти дані через кілька серверів або вузлів для обробки великого обсягу інформації.

MongoDB знаходить застосування у великій кількості проектів, включаючи веб-розробку, аналітику, системи управління контентом та інші області, де важлива гнучкість та швидкість відповіді [48].

В контексті архітектури даного конвеєру даних наявність OLTP бази даних надає можливість будувати звіти в реальному часі та є джерелом даних для інтерфейсу користувача навчальної платформи.

Для розгортання даної бази був MongoDB Atlas сервіс за допомогою Terraform.

В лістингу 3.3 наведено HQL код для розгортання MongoDB.

Лістинг 3.3 HQL код для розгортання MongoDB

```
resource "mongodbatlas_project" "project-student-scores" {
  name   = "student-scores"
  org_id = var.mongodbatlas_org_id
}

resource "mongodbatlas_cluster" "cluster-core" {
  project_id              = mongodbatlas_project.project-student-scores.id
  name                    = "cluster-core"

  # Provider Settings "block"
  provider_name = "TENANT"
  backing_provider_name = "AWS"
  provider_region_name = "US_EAST_1"
  provider_instance_size_name = "M0"
}
```

MongoDB Atlas – це хмарний сервіс управління базами даних MongoDB. Він надає можливість легко розгортати та керувати інфраструктурою MongoDB в області хмарних обчислень без потреби великих витрат часу та зусиль на адміністрування та моніторинг.

Основні характеристики MongoDB Atlas [48]:

– MongoDB Atlas працює на різних хмарних платформах, таких як Amazon Web Services (AWS), Microsoft Azure та Google Cloud Platform (GCP). Це дає користувачам можливість вибрати платформу, яка відповідає їхнім потребам та вимогам;

– за допомогою MongoDB Atlas можна легко створити кластер бази даних MongoDB всього кількома кліками в вебінтерфейсі. Сервіс автоматично налаштовує параметри та забезпечує можливість швидко розпочати роботу;

– сервіс забезпечує автоматичне резервне копіювання, моніторинг та поновлення. Крім того, він може автоматично виявляти та реагувати на відмови в системі для забезпечення стійкості роботи;

– MongoDB Atlas дозволяє розгортати кластери баз даних в різних регіонах світу, забезпечуючи глобальну доступність даних.

Підсумовуючи, AWS Lambda отримує повідомлення з потоку даних, трансформує їх, та зберігає до MongoDB. Код для AWS Lambda виконаний за допомогою мови програмування Java. В лістингу 3.4 наведено код головного методу лямбда функції:

Лістинг 3.4 Головний метод лямбда функції для обробки потоку оцінок студентів:

```
    @Override
    public Void handleRequest(KinesisEvent event, Context context) {
      MongoDatabase database = mongoClient.getDatabase(CORE_DB);
      MongoCollection<StudentScore> collection = database.getCollection(
        SCORE_COLLECTION,
        StudentScore.class
      );
      for (KinesisEventRecord record : event.getRecords()) {
        String rawScoreMessage = new String(record.getKinesis().getData().array());
        try {
          StudentScore studentScore = StudentScoreHelper.buildStudentScore(rawScoreMessage);
          collection.insertOne(studentScore);
```

```
            } catch (Exception e) {
                log.error("Error persisting student score", e);
                errorHandler.sendFailedMessageToSqs(rawScoreMessage);
            }
        }
        return null;
    }
```

Дану лямбда-функцію було розгорнуто за допомогою HQL-коду, представленого в лістингу 3.5.

Лістинг 3.5 HQL код для розгортання лямбда функції:

```
resource "aws_lambda_function" "kinesis_lambda_function" {
  filename         = "../../target/kinesis-lambda-1.0-SNAPSHOT.jar"
  function_name    = "kinesis-data-stream-reader-lambda"
  role             = aws_iam_role.lambda_iam_role.arn
  handler          = "ua.nure.lambda.java.KinesisHandler::handleRequest"
  runtime          = "java11"
  source_code_hash     =     filebase64sha256("../../target/kinesis-lambda-1.0-
SNAPSHOT.jar")
  memory_size = 2048
  timeout     = 900
  environment {
    variables = {
      MONGODB_CONNECTION_URI = var.mongo_connection_string,
      MONGODB_USER           = var.mongodb_username,
      MONGODB_PASSWORD       = var.mongodb_password,
    }
  }
}
```

В загальному, процес виглядає наступним чином:

– як тільки студент відправляє завдання на API Gateway, надходить новий запит з повідомлення про оцінку;

– повідомлення відправляється до потоку даних, розгорнутого за допомогою KDS;

– AWS Lambda отримує дані з потоку та зберігає їх в MongoDB.

На рисунку 3.3 зображено інтерфейс MongoDB Atlas який демонструє, що вхідні дані були успішно збережені в базі.

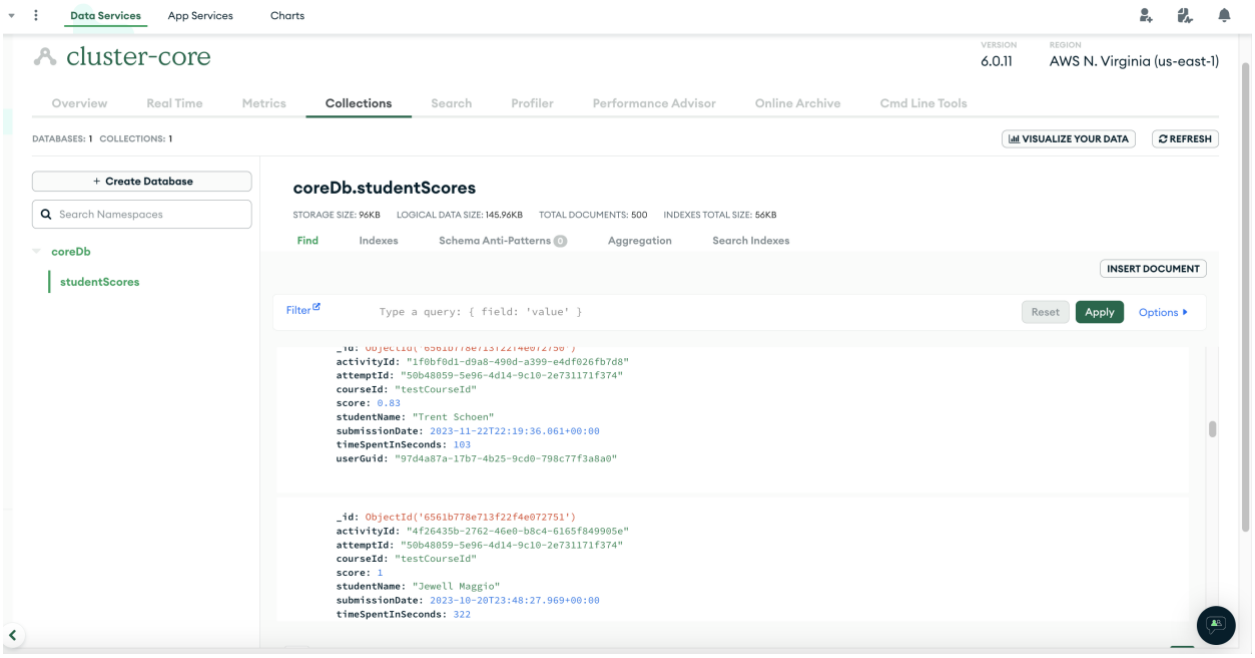

Рисунок 3.3 – Інтерфейс MongoDB Atlas

Останньою складовою конвеєру даних для приймання оцінок студентів є обробка помилок під час опрацювання повідомлень лямбда функцією. Це важлива частина конвеєру, яка допомагає гарантувати, що оцінки студентів не будуть втрачені в разі, якщо лямбда функція не може опрацювати повідомлення через внутрішню помилку або якщо база даних була недоступна протягом певного часу. AWS Lambda запускається для обробки набору записів із потоку даних та створює контрольну точку після успішної обробки кожного пакета. Це означає, що або весь пакет обробляється без

проблем, або весь пакет піддається повторному опрацюванню, доки не відбудеться успішна обробка, або записи видаляються з потоку через закінчення терміну зберігання даних. Поява шкідливого повідомлення, тобто такого, яке не може бути опрацьовано лямбдою через те, що воно не відповідає очікуваній схемі, призводить до збою обробки всього пакету. Це може призвести до дублювання результатів або затримки обробки даних і потенційної втрати даних.

AWS рекомендує два способи опрацювання подібних помилок. Перший варіант полягає в опрацюванні помилок у коді лямбда функції, шляхом перехоплення винятків та повернення успішного результату для обробки наступного пакету. Другий варіант полягає в застосуванні вбудованого механізму KDS для обробки помилок [18]. У даному конвеєрі був використаний перший варіант, процес якого зображений на рисунку 3.4.

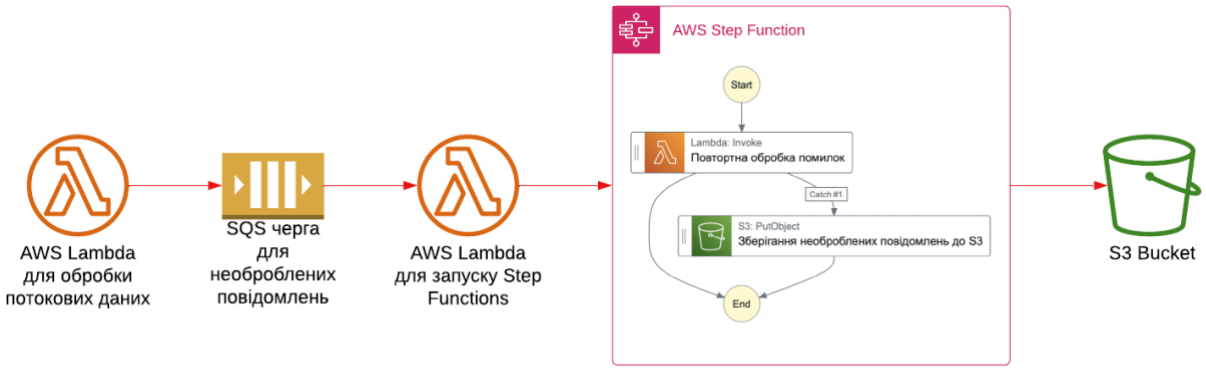

Рисунок 3.4 – Процес обробки помилок

Код лямбда-функції перехоплює помилку та відправляє отримане повідомлення до SQS (Simple Queue Service) черги. SQS дозволяє взаємодіяти різним компонентам розподіленої системи асинхронно за допомогою повідомлень. Потім окрема лямбда-функція читає повідомлення із черги та запускає процес повторного опрацювання повідомлення, який керується за допомогою AWS Step Functions. Даний сервіс дозволяє легко

координувати та керувати послідовностями кроків в різних AWS-сервісах, що є реалізацією моделі скінченного автомата [18].

AWS Step Functions отримує повідомлення, яке було необроблене основною лямбда-функцією, та запускає окрему лямбду, яка відповідальна за його повторне опрацювання. Step Functions надає можливість встановити кількість можливих повторних запусків лямбда-функції з виростанням експоненційної затримки, тобто різними проміжками часу між запусками. У випадку даного конвеєру даних було встановлено, що помилка може бути опрацьована 3 рази через 1, 5 та 25 хвилин між запусками відповідно. Це надає можливість автоматично опрацьовувати помилки. Наприклад, у ситуації, коли база даних була недоступною протягом 10 хвилин, то на 3-тю спробу повідомлення буде успішно оброблене та збережене до бази даних. Якщо повідомлення не було успішно оброблене за 3 спроби, то Step Functions зберігає його до S3 бакету. Потім розробники можуть переглянути зміст збереженого повідомлення та виявити причину помилки. Після усунення проблеми повідомлення може бути повторно відправлене до системи.

3.2 Розробка конвеєру даних для імпорту оцінок студентів до сховища даних

Наступним кроком є реалізація конвеєру даних для імпорту даних з MongoDB до Redshift, процес даного конвеєру представлений на рисунку 3.5.

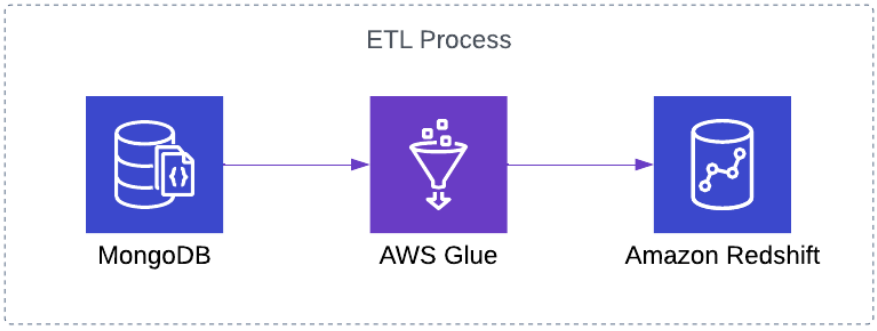

Рисунок 3.5 – Конвеєр даних для імпорту оцінок студентів до сховища даних

Redshift виконує роль OLAP (Online Analytical Processing) системи в даній архітектурі. OLAP – це підхід до обробки та аналізу даних, спрямований на виконання аналітичних операцій для отримання глибокого розуміння даних. OLAP використовується в сферах бізнесу та аналітики для швидкого та зручного доступу до великих даних з метою прийняття управлінських рішень [13].

В лістингу 3.6 наведено код для розгортання Redshift сервісу.

Лістинг 3.6 HQL код для розгортання Redshift сервісу:

```
resource "aws_redshiftserverless_namespace" "student_score_data" {
  namespace_name = "student-score-data"

  admin_username      = var.redshift_username
  admin_user_password = var.redshift_password
  db_name             = "student_scores"
}

resource "aws_redshiftserverless_workgroup" "student_score_data_workgroup" {
  namespace_name = aws_redshiftserverless_namespace.student_score_data.namespace_name
  workgroup_name = "student-score-data"
  security_group_ids = [aws_security_group.allow_all_traffic.id]
  publicly_accessible = true
  base_capacity       = 8
}
```

ETL процес імпорту даних з MongoDB до Redshift сховища даних здійснюється за допомогою сервісу AWS Glue. Даний сервіс був обраний для цього завдання, оскільки він надає наступні переваги [18]:

– AWS Glue – це Serverless сервіс, що означає, що користувачам не потрібно управляти інфраструктурою. Система автоматично масштабується відповідно до потреб завдання;

– сервіс інтегрується з іншими сервісами AWS, такими як Amazon S3, Amazon RDS, Amazon Redshift, забезпечуючи гнучкість у виборі джерел та призначень даних.

Код для розгортання AWS Glue джоби наведено в лістингу 3.7.

Лістинг 3.7 Код для розгортання AWS Glue джоби:

```
resource "aws_glue_job" "student-scores-export-job" {
  name     = "student-scores-export-job"
  role_arn = aws_iam_role.glue_iam_role.arn
  glue_version = "3.0"
  worker_type = "G.1X"
  timeout = 2
  number_of_workers = 2

  command {
    script_location = "s3://${aws_s3_bucket.glue-bucket.bucket}/scripts/glue-job.scala"
  }

  default_arguments = {
    "--job-language" = "scala"
    "--class" = "GlueApp"
    "--enable-glue-datacatalog" = "true",
    "--enable-continuous-cloudwatch-log" = "true"
    "--TempDir" = local.aws_glue_temp_dir
  }
}
```

AWS Glue джоби працюють на основі Apache Spark та підтримують мову програмування Scala. Для створення джоби програмний код повинен бути завантажений до S3 бакету та створені з'єднання.

На рисунку 3.6 наведено успішне виконання джоби в AWS консолі.

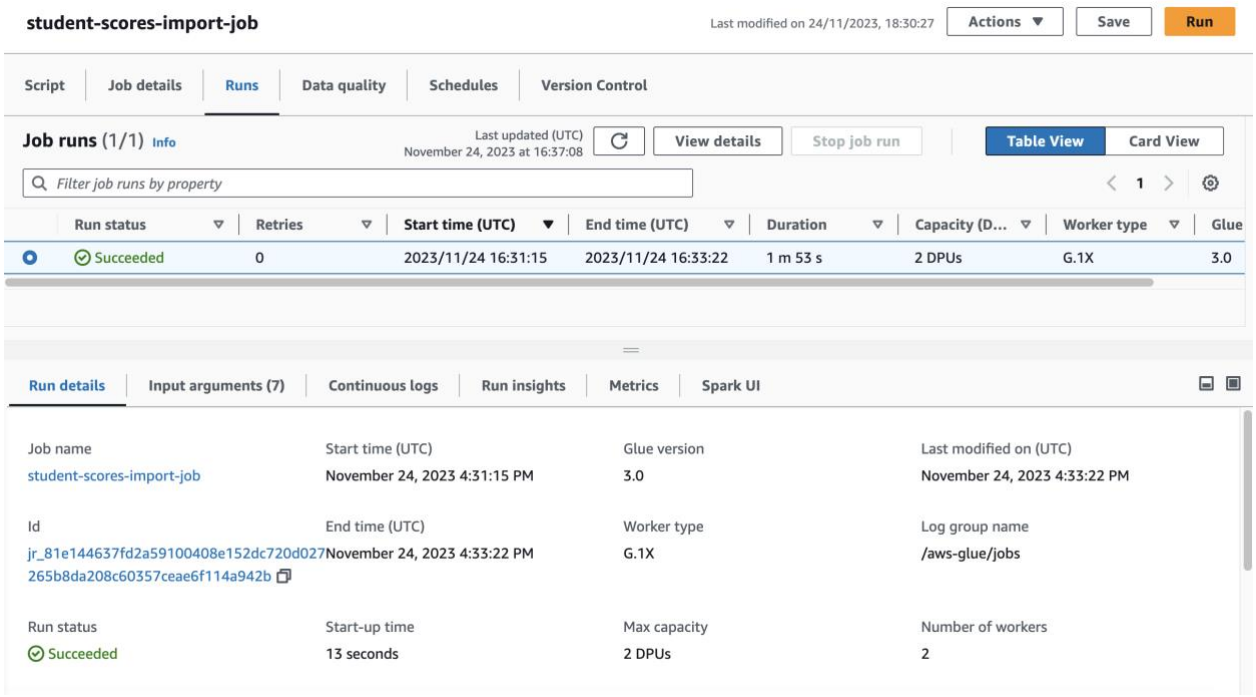

Рисунок 3.6 – AWS Glue консоль

Код джоби представлено в лістингу 3.8.

Лістинг 3.8 Код AWS Glue джоби реалізований за допомогою мови програмування Scala:

```
object GlueApp {
  def main(sysArgs: Array[String]) {
    val spark: SparkContext = new SparkContext()
    val glueContext: GlueContext = new GlueContext(spark)
    val args = GlueArgParser.getResolvedOptions(sysArgs, Seq("JOB_NAME").toArray)
    Job.init(args("JOB_NAME"), glueContext, args.asJava)
    val mongodb_options = JsonOptions(
```

```
      s"""{
        |    "uri": "${Properties.mongoUri}",
        |    "database": "coreDb",
        |    "collection": "studentScores",
        |    "username": "${Properties.userName}",
        |    "password": "${Properties.password}"
        | }""".stripMargin)
      val mongoDbDataFrame = glueContext.getSource(
       connectionType = "mongodb",
       connectionOptions = mongodb_options
      ).getDynamicFrame()
      val updatedDataFrame = mongoDbDataFrame.applyMapping(
       mappings = Properties.mappings,
       caseSensitive = false
      )
      val redshiftOptions = JsonOptions(
       s"""{
        |  "redshiftTmpDir": "${args("TempDir")}",
        |  "useConnectionProperties": "true",
        |  "dbtable": "public.student_scores",
        |  "connectionName": "redshift_connection",
        |  "preactions":  "${Properties.query}"
        |}""".stripMargin)
      glueContext.getSink(connectionType = "redshift", connectionOptions =
redshiftOptions)
       .writeDynamicFrame(updatedDataFrame)
      Job.commit()
    }
}
```

Результатом роботи джоби є завантажені дані до Redshift, на рисунку 3.7 зображено інтерфейс Redshift, який демонструє успішну роботу AWS Glue джоби.

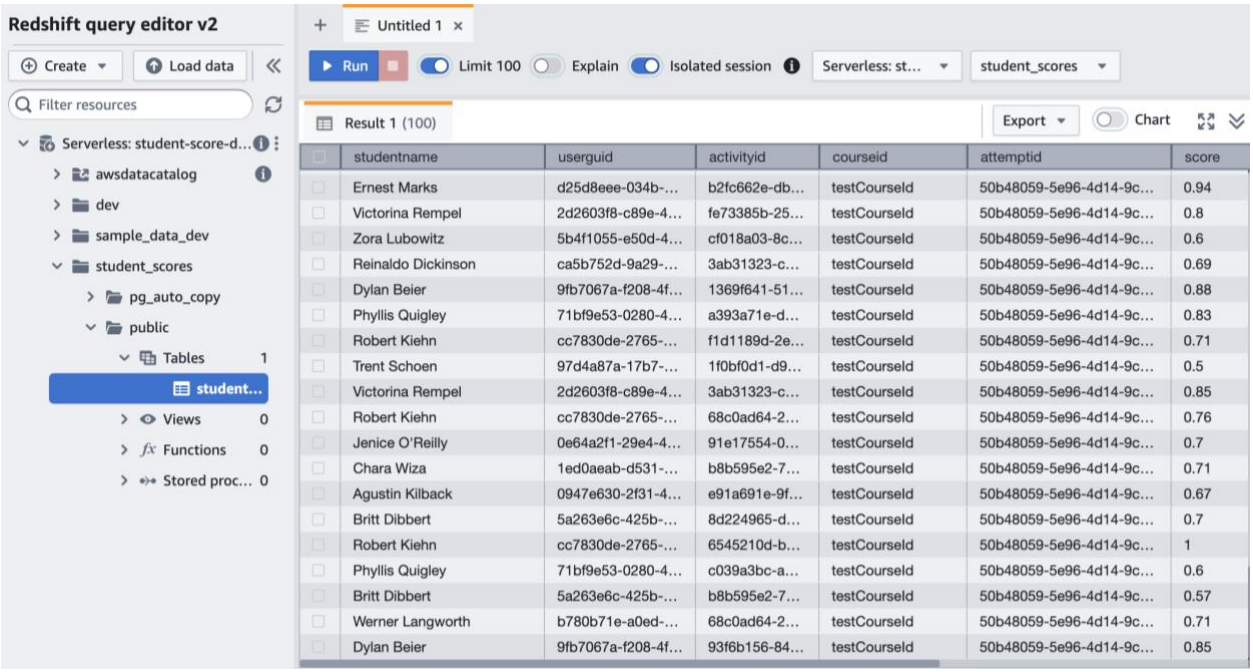

Рисунок 3.7 – Інтерфейс Amazon Redshift

Таким чином, за допомогою даного конвеєру даних був реалізований ETL процес експорту оцінок студентів з MongoDB до Amazon Redshift для їх історичного зберігання, аналізу та побудови звітів.

3.3 Розробка звітів про успішність студентів

Останнім елементом загального конвеєру даних, який складається з двох попередньо розроблених конвеєрів для поглинання оцінок студентів та їх експорту до сховища даних, є візуалізація та аналіз даних.

Архітектура загального конвеєру, де всі компоненти представлені разом, зображена на рисунку 3.8.

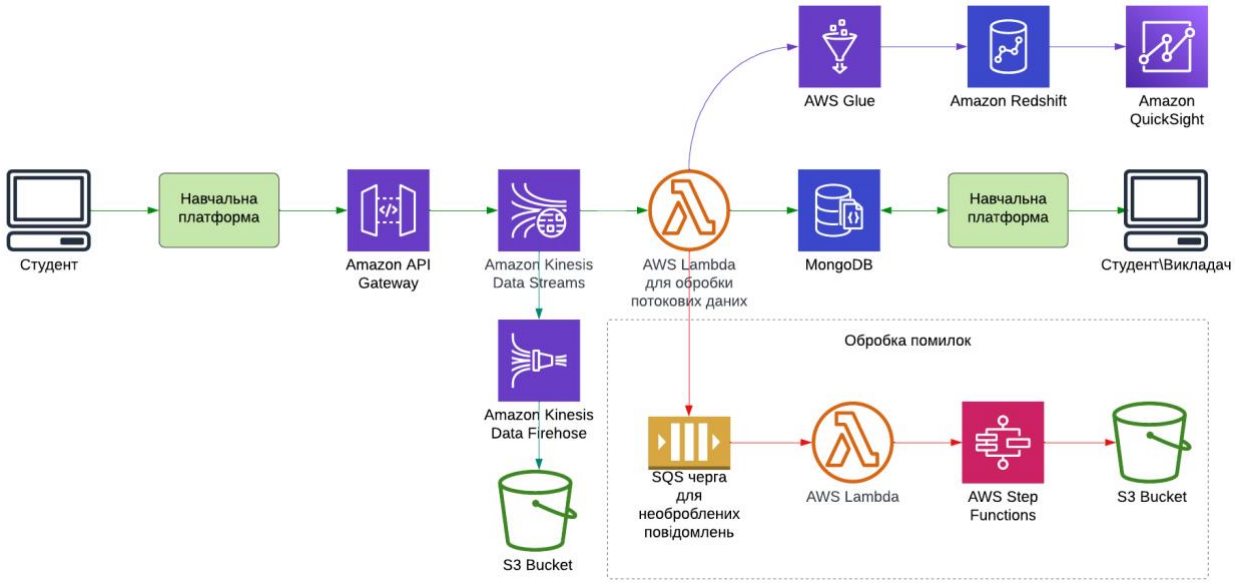

Рисунок 3.8 – Архітектура загального конвеєру даних

Виходячи з представленої архітектури, останнім компонентом конвеєру, який буде розглянуто, є Amazon QuickSight. Він відповідальний за візуалізацію, аналіз даних та побудову звітів. Сервіс автоматично інтегрується з іншими сервісами AWS, такими як Amazon S3, Amazon RDS, Amazon Redshift. Це робить можливим ефективне використання та аналіз даних, які зберігаються в AWS. Також однією із переваг цього сервісу є можливість вбудовування звітів безпосередньо в вебзастосунок за допомогою Amazon QuickSight Embedding SDK (Software Development Kit) [18]. Тобто звіти розроблені за допомогою інструментів QuickSight можуть бути відображені для студентів та викладачів у вебзастосунку навчальної платформи.

Для побудови звітів, в першу чергу, потрібно підготувати тестові дані. Для цього був розроблений код за допомогою мови програмування JavaScript, який генерує тіло повідомлення та відправляє його на кінцеву точку (endpoint) API Gateway, яка була підготовлена раніше. Загалом було згенеровано та збережено дані в MongoDB для тестового курсу, в який зараховано 25 студентів. Потім дані з MongoDB були імпортовані до Amazon

Redshift, що дає змогу використовувати Amazon QuickSight для побудови звітів.

Потім Amazon QuickSight був підключений до попередньо створеної бази в Amazon Redshift для зберігання оцінок. На рисунку 3.9 зображено інтерфейс консолі Amazon QuickSight, що позначає успішне з'єднання з Redshift базою.

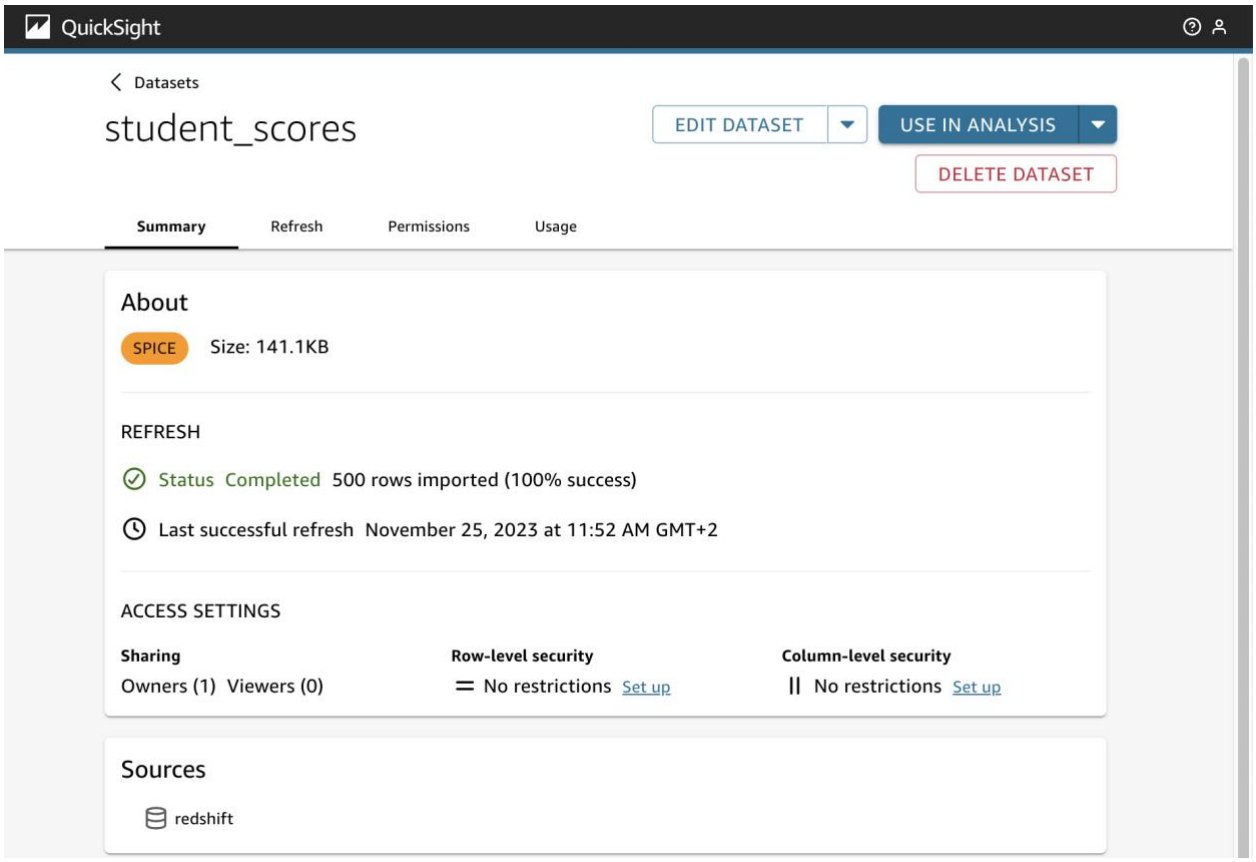

Рисунок 3.9 – Інтерфейс Amazon QuickSight

Наступним етапом є створення звітів, який полягає в визначенні вимог та підготовці даних у відповідному форматі. Amazon QuickSight надає широкий набір вбудованих функцій обчислення, що дозволяють користувачам виконувати різноманітні операції обробки та агрегації даних для побудови звітів. Ці функції можна використовувати для розрахунків на числових, рядкових, дата-часових, географічних та інших типах даних. Основні категорії вбудованих функцій обчислення включають. Ці вбудовані

можливості були використані для розрахунку нових значень, таких як середній час, витрачений на проходження курсу студентом, та визначення ECTS (European Credit Transfer and Accumulation System) оцінок.

На рисунку 3.10 наведено інтерфейс Amazon QuickSight для створення звітів.

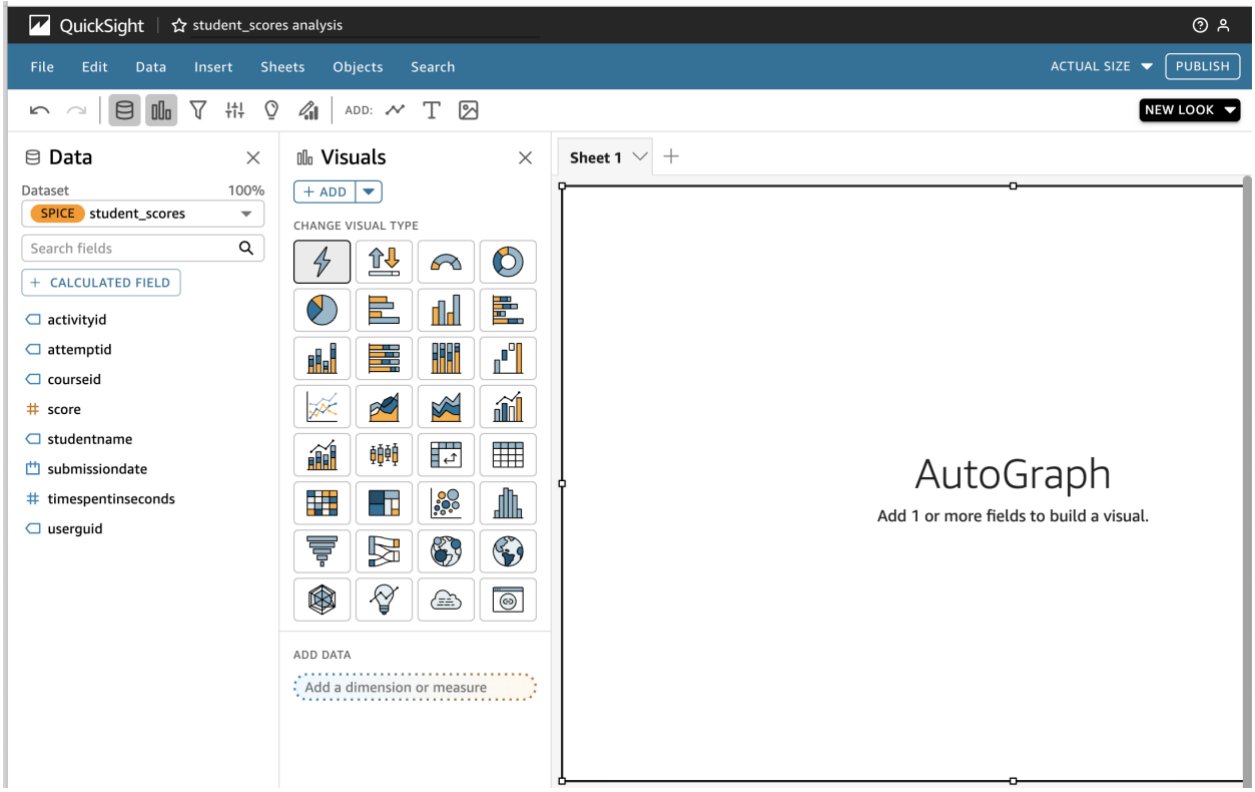

Рисунок 3.10 – Інтерфейс Amazon QuickSight для створення звітів

На основі даних про оцінки студентів, які зберігаються в Amazon Redshift, було побудовано декілька звітів, які націлені допомагти викладачам та адміністраціям навчальних закладів оцінити успішність студентів.

На рисунку 3.11 зображено звіт про успішність студентів при проходженні початкового курсу, який включає в себе наступні показники:

– оцінка (score). Показує середню оцінку певного студента за проходження курсу;

– час затрачений на проходження курсу в хвилинах (time spent in minutes). Даний показник представляю собою загальний час, витрачений студентом на проходження курсу;

– кількість виконаних завдань (number of completed activities). Позначає, скільки завдань було виконано студентом в рамках курсу;

– останній час відправки завдання (last submission date). Дає змогу зрозуміти, коли студент відправив завдання в рамках курсу останній раз.

Student Performance Report

Name	Score	Time Spent In Minutes	Number of completed activities	Last Submission Date
Jason Kuphal	87.57%	84.4	14	Oct 3, 2023 9:29am
Jewell Maggio	86.3%	35.72	10	Oct 5, 2023 8:07am
Agustin Kilback	83.75%	102.67	20	Oct 7, 2023 4:51am
Dana Gottlieb	82.67%	114.73	21	Oct 4, 2023 8:02am
Victorina Rempel	82.24%	131.1	25	Oct 1, 2023 3:28am
Marci Dare	81.32%	101.1	19	Oct 8, 2023 11:11am
Ron Jacobi	79.8%	110.77	25	Oct 6, 2023 4:19am
Magda Morar	79.22%	106.7	18	Oct 4, 2023 8:43am
Kandace Zemlak	78.36%	117.45	22	Oct 5, 2023 6:01am
Phyllis Quigley	77.68%	138.13	28	Oct 1, 2023 7:56am
Britt Dibbert	77.62%	100.65	21	Oct 2, 2023 6:57pm
Major Grimes	77.41%	107.1	22	Oct 8, 2023 3:22am
Deshawn Braun	77%	108.58	27	Oct 2, 2023 2:51am
Reinaldo Dickinson	76.55%	26.68	11	Oct 3, 2023 1:48pm
Zora Lubowitz	75.8%	103.87	20	Oct 4, 2023 1:04pm
Trent Schoen	75.54%	115.63	26	Sep 30, 2023 11:40pm
Ernest Marks	75.23%	93.48	22	Oct 1, 2023 12:12am

Рисунок 3.11 – Звіт про успішність студентів проходження начального курсу

Також на рисунку представлений звіт, який складається з декількох графічних елементів, кожен з яких відображає окремий показник (рис. 3.12). Розглянемо більш детально кожний із елементів:

– середня оцінка за курс (class average score). Показує середню оцінку за курс між всіма студентами курсу;

– середній час, витрачений на проходження курсу в хвилинах (average time spent to take the course in minutes). Представляє значення, яке вказує на те, скільки часу студенти витратили на проходження в середньому;

– оцінки студентів (student scores). Графічно відображає оцінки всіх студентів в курсі, що дає змогу візуально оцінити успішність студентів;

– розподіл ECTS оцінок за курс (student overall course grades). Показує кількість студентів, які отримали певну ECTS оцінку;

– розподіл ECTS оцінок за виконані завдання (activity grade distribution). Відображає, скільки завдань було виконано на певну ECTS оцінку.

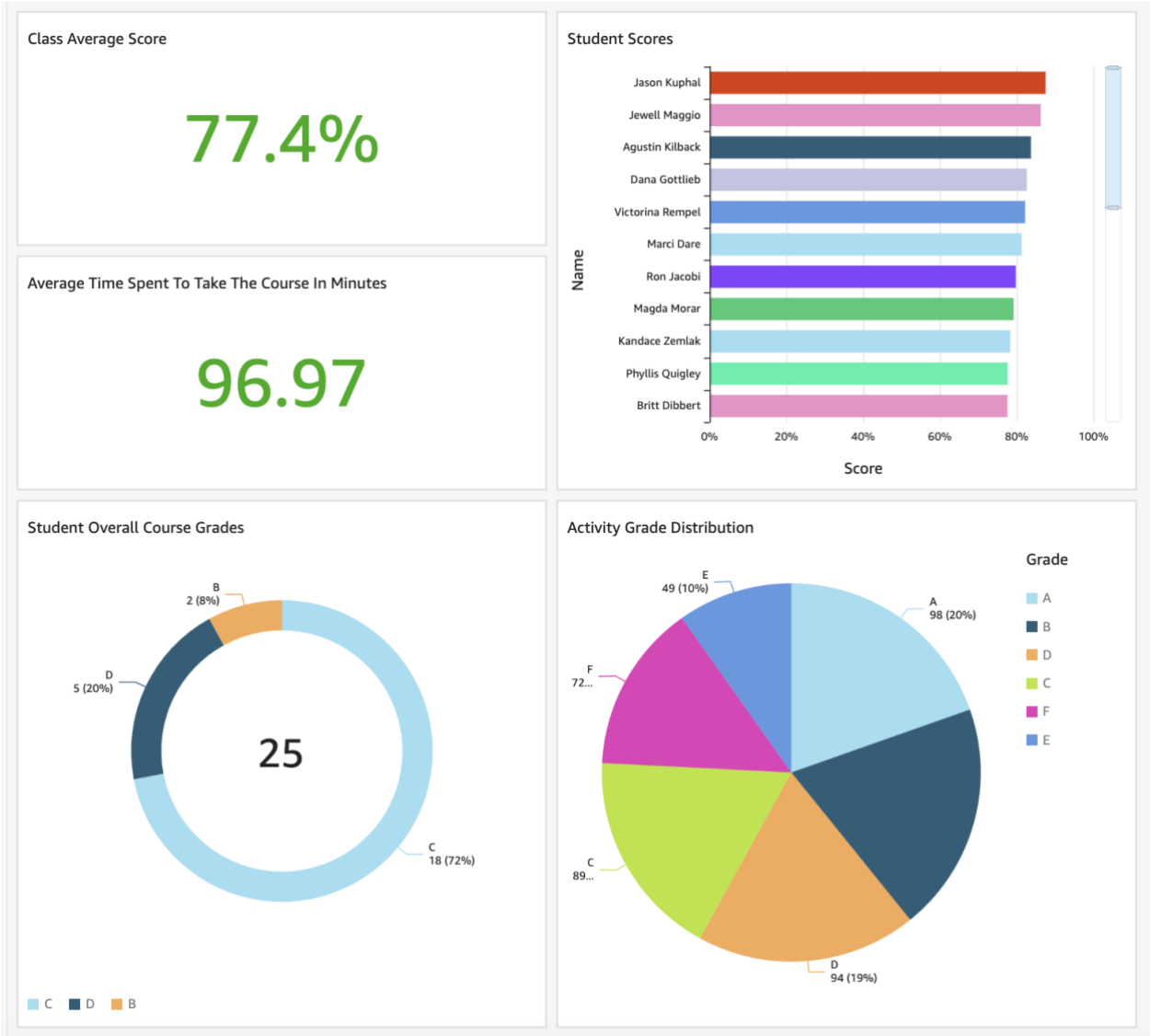

Рисунок 3.12 – Комплексний звіт показників успішності

Аналіз такого звіту дозволяє легко оцінити успішність проходження курсу. Наприклад, 18 студентів отримали оцінку C, і тільки 2 студента отримали оцінку B, при відсутності оцінок A. Середня оцінка за курс склала 77%, а середній час проходження 97 хвилин. Дані показники говорять про те, що всі студенти успішно пройшли курс, але середня оцінка недостатньо висока, тому надалі доцільно провести аналіз того, як можна покращити дані показники.

Таким чином, було розроблено конвеєр даних для поглинання оцінок студентів та його централізованого зберігання. Також були використані інструменти візуалізації для побудови звітів. Конвеєр даних був апробований на заздалегідь підготовлених даних. Оскільки для виконання завдання використовувалися хмарні сервіси AWS, які працюють за моделлю Serverless, пропускна здатність конвеєру даних може автоматично масштабуватися відповідно до поточного навантаження. Це дозволяє використовувати даний процес для опрацювання big data.

ВИСНОВКИ

В ході виконання кваліфікаційної роботи проаналізована актуальність застосування хмарних технологій для обробки великих даних, розглянуто процес побудови конвеєру даних, розроблена та досліджена його реалізація для поглинання оцінок студентів та побудови звітів про успішність проходження навчального курсу.

Визначено, що конвеєр даних – це система для автоматизованого збору, обробки, трансформації та передачі інформації від одного етапу обробки до іншого, та складається з наступних компонентів: джерело даних, поглинання, зберігання, обробка та візуалізація.

Було проведено порівняння постачальників хмарних послуг, і був обраний AWS для виконання завдання, оскільки це найпопулярніший провайдер на сьогоднішній день.

Розглянуто основні методи роботи з big data за допомогою хмарних сервісів AWS, зокрема для збору, зберігання та аналізу даних.

На основі зібраних теоретичних відомостей розроблено та вивчено конвеєр даних для збору оцінок студентів та їх централізованого зберігання. Використано інструменти візуалізації даних для створення звітів. Розробка була протестована на тестових даних про оцінки студентів. Оскільки для виконання завдання використовувалися хмарні сервіси AWS, які не потребують налаштування інфраструктури з боку користувачів, то пропускна здатність конвеєру може автоматично масштабуватися згідно з поточним навантаженням, що дозволяє охарактеризувати дану розробку саме як big data конвеєр даних.

Результати роботи апробовано у вигляді тез доповідей під час International Scientific And Practical Conference «Current trends in the development of youth theories» [**Error! Reference source not found.**].

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Upadhyay, S., Manwani, R., Varshney, S., & Jain, S. (2021). Analytics and Storage of Big Data. In ISIC (pp. 202-210).

2. Андрощук, О., Головченко, О., Литовченко, Г., & Петрушен, М. (2021). Аналіз поняття хмарні технології: види, категорії, переваги та недоліки. Молодий вчений, (6 (94)), 83-87.

3. Горбачук, В., Гавриленко, С., Голоцуков, Г., & Дунаєвський, М. (2020). Засади розвитку хмарних технологій.

4. Маркова, О. М., Семеріков, С. О., & Стрюк, А. М. (2015). Хмарні технології навчання: витоки. Інформаційні технології і засоби навчання, (46, вип. 2), 29-44.

5. Кулижко, А. О. (2021). Застосування Data pipelines в хмарних сервісах для медіа–аналітики засобів масової інформації України (Bachelor's thesis, КПІ ім. Ігоря Сікорського).

6. Romaniuk, P. I. (2023). Хмарні технології: аналіз, перспективи, реалізації. COMPUTER-INTEGRATED TECHNOLOGIES: EDUCATION, SCIENCE, PRODUCTION, (50), 108-113.

7. Mell P. M., Grance T. (2011). The NIST definition of cloud computing. Gaithersburg, MD : National Institute of Standards and Technology.

8. Erl, T., Puttini, R., & Mahmood, Z. (2013). Cloud computing: concepts, technology & architecture. Pearson Education.

9. Alwan, H. B., & Ku-Mahamud, K. R. (2020, February). Big data: Definition, characteristics, life cycle, applications, and challenges. In IOP Conference Series: Materials Science and Engineering (Vol. 769, No. 1, p. 012007). IOP Publishing.

10. Oleghe, O., & Salonitis, K. (2020). A framework for designing data pipelines for manufacturing systems. Procedia CIRP, 93, 724-729.

11. Самойленко, Б. (2023). Архітектура data pipeline для збору даних про активність користувачів.

12. Oussous, A., Benjelloun, F. Z., Lahcen, A. A., & Belfkih, S. (2018). Big Data technologies: A survey. Journal of King Saud University-Computer and Information Sciences, 30(4), 431-448.

13. Kleppmann, M. (2019). Designing Data-Intensive Applications.

14. Srivastava, P., & Khan, R. (2018). A review paper on cloud computing. *International Journal of Advanced Research in Computer Science and Software Engineering*, *8*(6), 17-20.

15. Puthal, D., Sahoo, B. P., Mishra, S., & Swain, S. (2015, January). Cloud computing features, issues, and challenges: a big picture. In *2015 International conference on computational intelligence and networks* (pp. 116-123). IEEE.

16. Gupta, B., Mittal, P., & Mufti, T. (2021, March). A review on amazon web service (aws), microsoft azure & google cloud platform (gcp) services. In *Proceedings of the 2nd International Conference on ICT for Digital, Smart, and Sustainable Development, ICIDSSD 2020, 27-28 February 2020, Jamia Hamdard, New Delhi, India*.

17. Lu, T., Hoyer, S., Wang, Q., Hu, L., & Chen, Y. F. (2021, July). Distributed data processing for large-scale simulations on cloud. In *2021 IEEE International Joint EMC/SI/PI and EMC Europe Symposium* (pp. 53-58). IEEE.

18. AWS Documentation – URL: https://docs.aws.amazon.com/ (дата звернення 16.10.2023).

19. Azure Documentation – URL: https://learn.microsoft.com/ (дата звернення 18.10.2023).

20. GCP Documentation – URL: https://cloud.google.com/docs/ (дата звернення 19.10.2023).

21. Wankhede, P., Talati, M., & Chinchamalatpure, R. (2020). Comparative study of cloud platforms-microsoft azure, google cloud platform and amazon EC2. *J. Res. Eng. Appl. Sci*, *5*(02), 60-64.

22. Sailakshmi, V. (2021). Analysis of Cloud Security Controls in AWS, Azure, and Google Cloud.

23. Bansal, A., Jain, R., & Modi, K. (2019). Big data streaming with spark. Big Data Processing Using Spark in Cloud, 23-50.

24. Pandis, I. (2021). The evolution of Amazon redshift. Proceedings of the VLDB Endowment, 14(12), 3162-3174.

25. Chapin, J., & Roberts, M. (2020). Programming AWS Lambda: build and deploy serverless applications with Java. O'Reilly Media.

26. Гороховатський В.О., Гадецька С.В., Стяглик Н.І. (2019) Вивчення статистичних властивостей моделі блочного подання для множини дескрипторів ключових точок зображень. Радіоелектроніка, інформатика, управління, № 2, С. 100–107.

27. Gadetska, S. V., & Gorokhovatskyi, V. O. (2018). Statistical measures for computation of the image relevance of visual objects in the structural image classification methods. Telecommunications and Radio Engineering, 77(12), pp. 1041– 1053.

28. Gorokhovatskyi, V., & Vlasenko, N. (2021). Редукція опису зображення у складі множини дескрипторів на основі метричного критерію інформативності. Advanced Information Systems, 5(4), 10-16.

29. Gadetska, S., Gorokhovatskyi, V., Stiahlyk, N., & Vlasenko, N. (2022). Aggregate Parametric Representation of Image Structural Description in Statistical Classification Methods.

30. Gorokhovatskyi, O., Peredrii, O., Gorokhovatskyi, V., & Vlasenko, N. (2023). Explanation of CNN image classifiers with hiding parts. In Explainable Deep Learning AI (pp. 125-146). Academic Press.

31. Гороховатський В., Творошенко І., Сидоренко Д. (2021) Класифікація зображень із використанням кластерного подання, Міжн. науковий симпозіум «Інтелектуальні рішення-С». Обчислювальний інтелект (результати, проблеми, перспективи). Теорія прийняття рішень: праці міжн. наук. симпозіуму (Вересень 29, 2021). Київ – Ужгород, С. 44-45.

32. Гороховатський, В. О., Передрій, О. О., Творошенко, І. С., & Марков, Т. Є. (2023). Матриця відстаней для множини компонентів

структурного опису як інструмент для створення класифікатора зображень. С. 5–13.

33. Gorokhovatskyi V.A. (2018) Image Classification Methods in the Space of Descriptions in the Form of a Set of the Key Point Descriptors. Telecommunications and Radio Engineering, 77 (9), pp. 787-797.

34. Gorokhovatsky, V.O. and Gadetska, S.V., (2019) Determination of Relevance of Visual Object Images by Application of Statistical Analysis of Regarding Fragment Representation of their Descriptions, Telecommunications and Radio Engineering, 78 (3), pp. 211–220.

35. Gorokhovatsky V.A. (2016). Efficient Estimation of Visual Object Relevance during Recognition through their Vector Descriptions. Telecommunications and Radio Engineering. – Vol. 75, No 14. – P. 1271–1283.

36. Gorokhovatskiy,V.A. (2011). Compression of Descriptions in the Structural Image Recognition.Telecommunications and Radio Engineering. –, Vol. 70, No 15. – P. 1363–1371.

37. Pomazan V., Tvoroshenko I., Gorokhovatskyi V. (2023) Handwritten character recognition models based on convolutional neural networks, International Journal of Academic Engineering Research, 7(9), pp. 64-72.

38. Tvoroshenko I., Gorokhovatskyi V., Kobylin O., and Tvoroshenko A. (2023) Application of deep learning methods for recognizing and classifying culinary dishes in images, International Journal of Academic and Applied Research, 7(9), pp. 57-70.

39. Gorokhovatskyi V., Tvoroshenko I. (2023) Identification of visual objects by the search request. International scientific symposium «INTELLIGENT SOLUTIONS-S». Computational intelligence (results, problems and perspectives). Decision making theory: proceedings of the international symposium, September 28, 2023, KyivUzhorod, Ukraine, pp. 25-27.

40. Gadetska, S.V., Gorokhovatskyi, V.O., Stiahlyk, N.I., Vlasenko, N.V. (2021). Statistical data analysis tools in image classification methods based on the

description as a set of binary descriptors of key points. Radio Electronics, Computer Science, Control,, №4, pp. 58-68.

41. Pomazan V., Tvoroshenko I., and Gorokhovatskyi V. (2023) Development of an application for recognizing emotions using convolutional neural networks, International Journal of Academic Information Systems Research, 7(7), pp. 25-36.

42. Gorokhovatskyi, O., Gorokhovatskyi, V., & Peredrii, O. (2018). Analysis of application of cluster descriptions in space of characteristic image features. Data, 3(4), 52.

43. Гороховатський, В. А. (2014). Структурний аналіз та інтелектуальна обробка даних у комп'ютерному зорі, моногр., 316 с.

44. Yakovleva O., and Nikolaieva K. (2020) Research of descriptor based image normalization and comparative analysis of SURF, SIFT, BRISK, ORB, KAZE, AKAZE descriptors. Advanced Information Systems, 4 (4), pp. 89-101.

45. Тимчишин, Р. М., Волков, О. Є., Господарчук, О. Ю., & Богачук, Ю. П. (2018). Сучасні підходи до розв'язання задач комп'ютерного зору. Управляючі системи і машини. – № 6. – С. 46-73.

46. Apache Spark Documentation – URL: https://spark.apache.org/docs/latest/ (дата звернення 22.10.2023).

47. Terraform Documentation – URL: https://developer.hashicorp.com/terraform/docs (дата звернення 30.10.2023).

48. MongoDB Documentation – URL: https://www.mongodb.com/docs/ (дата звернення 02.11.2023).