Лабораторна робота № 4. Пошукові СУБД: Elasticsearch

МЕТА РОБОТИ: отримати навички роботи з пошуковими базами даних.

ПОСТАНОВКА ЗАДАЧІ

Додати до застосунку, який був розроблений в попередній лабораторній роботі (що була присвячена графовим СУБД) логування роботи застосунку. Логи записувати безпосередньо до пошукової СУБД Elasticsearch. Для аналізу логів використовувати Kibana.

ПОРЯДОК ВИКОНАННЯ

ЧАСТИНА 1 (обов'язкова, оцінюється у **5 балів**).

1. Встановити СУБД Elasticsearch з офіційного сайту https://www.elastic.co/downloads/elasticsearch.

Порядок установки СУБД Elasticsearch на ОС Windows:

1.1. Завантажте ZIP-архів: Виберіть версію для Windows (зазвичай це ZIP-архів).

1.2. Створіть нову папку для Elasticsearch у зручному місці, наприклад, C:\elasticsearch.

1.3. Розпакуйте вміст завантаженого ZIP-архіву в цю папку. У C:\elasticsearch у вас з'явиться папка з ім'ям версії (наприклад, elasticsearch-9.x.x).

1.4. Перейдіть до папки конфігурації C:\elasticsearch\elasticsearch-9.x.x\config.

1.5. Відредагуйте **elasticsearch.yml під локальну розробку**:

1.5.1. Відкрийте elasticsearch.yml у текстовому редакторі (наприклад, Notepad++ або VS Code).

1.5.2. Знайдіть рядок network.host: 192.168.0.1 (або закоментований). Змініть його на: `network.host: localhost`

1.5.3. Знайдіть рядок `http.port: 9200` (порт за замовчуванням). Розкомментуйте цей рядок.

1.5.4. Частково відключіть безпеку (тільки для локальної розробки/навчання, не для продакшена!):

```
xpack.security.enabled: true
xpack.security.enrollment.enabled: true
xpack.security.http.ssl.enabled: false
xpack.security.transport.ssl.enabled: false
```

УВАГА ! Відключення безпеки робить ваш Elasticsearch вразливим. Ніколи не робіть цього у продакшені. Для навчання це може бути прийнятним для спрощення.

1.5.5. Збережіть зміни у файлі.

1.6. Запустіть Elasticsearch:

1.6.1. Відкрийте командний рядок (CMD або PowerShell) від імені адміністратора.

1.6.2. Перейдіть до папки bin Elasticsearch:

```
cd C:\elasticsearch\elasticsearch-9.x.x\bin
```

1.6.3. Запустіть на виконання bat-файл:

```
elasticsearch.bat
```

Elasticsearch почне запускатися. Це може тривати деякий час. Ви побачите багато логів у консолі.

1.7. Встановіть паролі для вбудованих користувачів Elasticserch:

1.7.1. Відкрийте НОВЕ вікно командного рядка (CMD або PowerShell) у тій же директорії від імені адміністратора (C:\elasticsearch\elasticsearch-9.x.x\bin).

1.7.2. Виконайте команду для встановлення/заміни паролів:

```
elasticsearch-setup-passwords auto
```

або

```
elasticsearch-reset-password
```

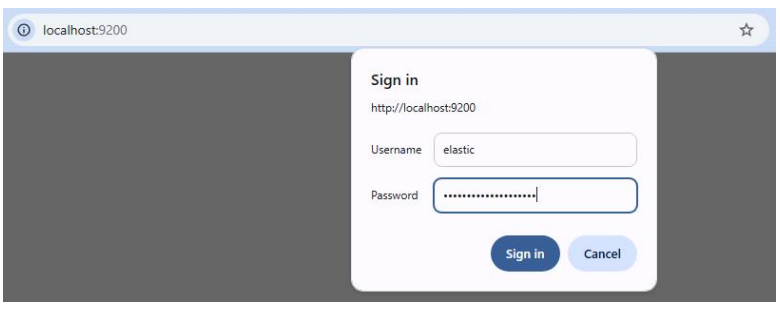
Ця команда автоматично згенерує та встановить паролі для вбудованих користувачів (таких як elastic, kibana_system, logstash_system тощо).

ОБОВ'ЯЗКОВО ЗАПИШІТЬ ВСІ ЗГЕНЕРОВАНІ ПАРОЛІ! Особливо паролі для elastic та kibana_system. Вам вони знадобляться.

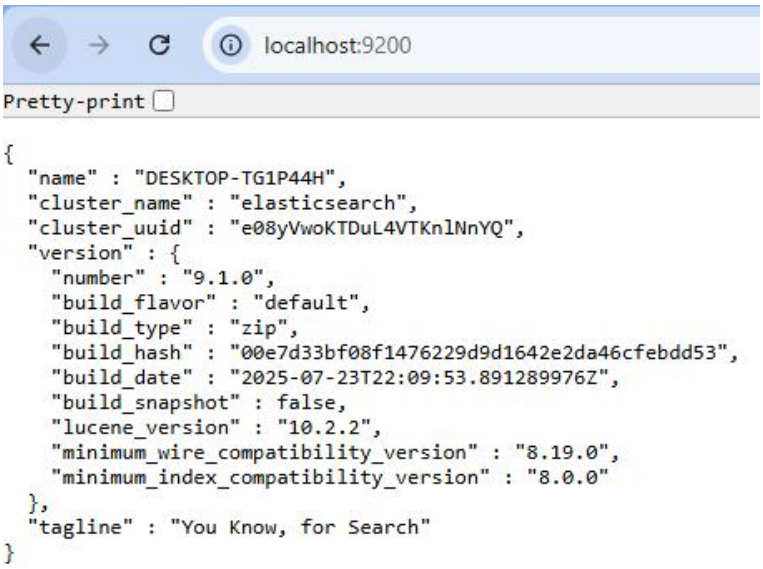
1.8. Перевірка роботи Elasticsearch

1.8.1. Відкрийте веб-браузер.

1.8.2. Перейдіть за адресою: http://localhost:9200. Введіть згенерований на попередньому кроці пароль для користувача `elastic`.



1.8.3. Ви повинні побачити JSON-відповідь з інформацією про ваш кластер Elasticsearch (ім'я, версія тощо).



2. Встановити застосунок Kibana для роботи з даними (здійснення запитів, аналітики, візуалізації тощо), що зберігаються в Elasticsearch, з офіційного сайту https://www.elastic.co/downloads/kibana:

2.1. Завантажте ZIP-архів для Windows (зазвичай це файл на кшталт kibana-<version>-windows-x86_64.zip

2.2. Створіть нову папку у зручному для вас місці, наприклад: C:\kibana.

2.3. Розпакуйте завантажений ZIP-архів Kibana до цієї нової папки.

2.4. Перейдіть до папки конфігурації C:\kibana\kibana-9.x.x\config.

2.5. Відредагуйте **kibana.yml під локальну розробку**:

2.5.1. Відкрийте kibana.yml у текстовому редакторі (наприклад, Notepad++ або VS Code).

2.5.2. Знайдіть і відредагуйте (або розкоментуйте, видаливши # на початку рядка) наступні рядки:

```
server.port: 5601
server.host: "localhost"
elasticsearch.hosts: ["http://localhost:9200"]
elasticsearch.username: "kibana_system"
elasticsearch.password: "your_password"
```

“your_password” змініть на той пароль, який було згенеровано у пункті 1.7.2.

2.5.3. Збережіть зміни у файлі.

2.6. Запустіть Kibana:

2.6.1. Відкрийте командний рядок (CMD або PowerShell) від імені адміністратора.

2.6.2. Перейдіть до папки bin Kibana:

```
cd C:\kibana\kibana-9.x.x\bin
```

2.6.3. Запустіть на виконання bat-файл:

```
kibana.bat
```

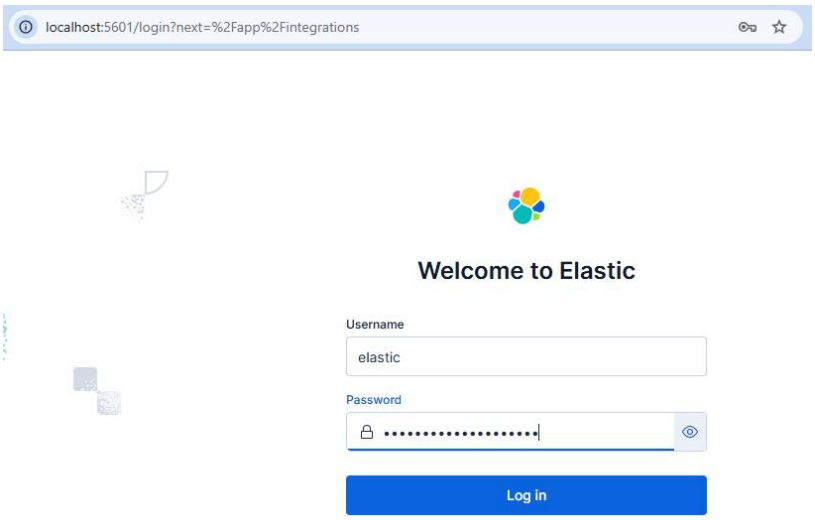
Kibana почне запускатися. Це може зайняти деякий час. Ви побачите вивід логів у командному рядку. Дочекайтеся повідомлення, що Kibana запущена.

2.7. Перевірте доступ до Kibana.

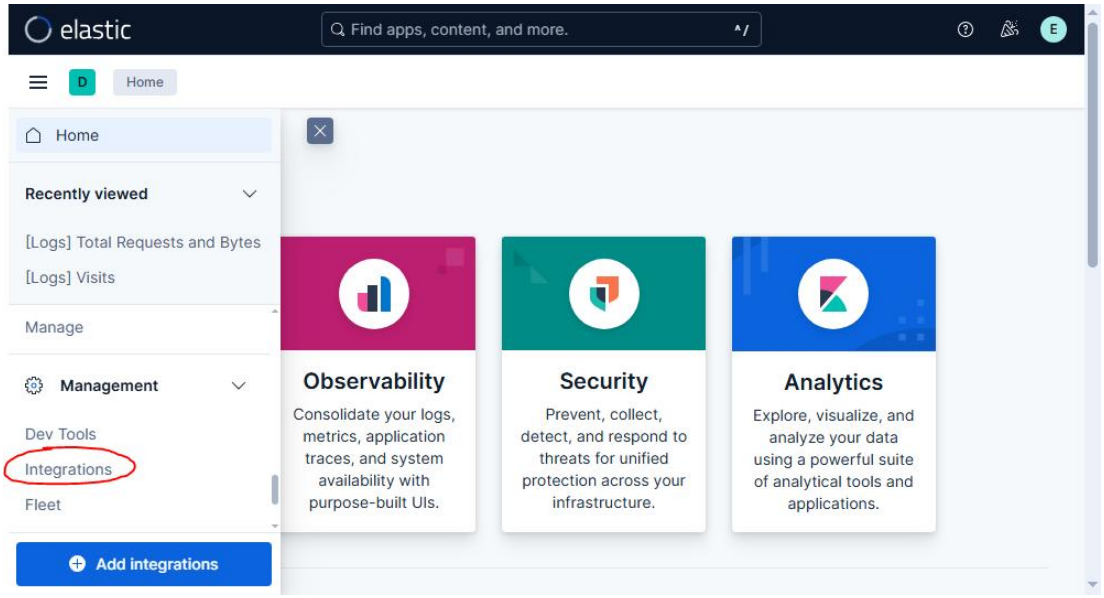
2.7.1. Відкрийте ваш веб-браузер.

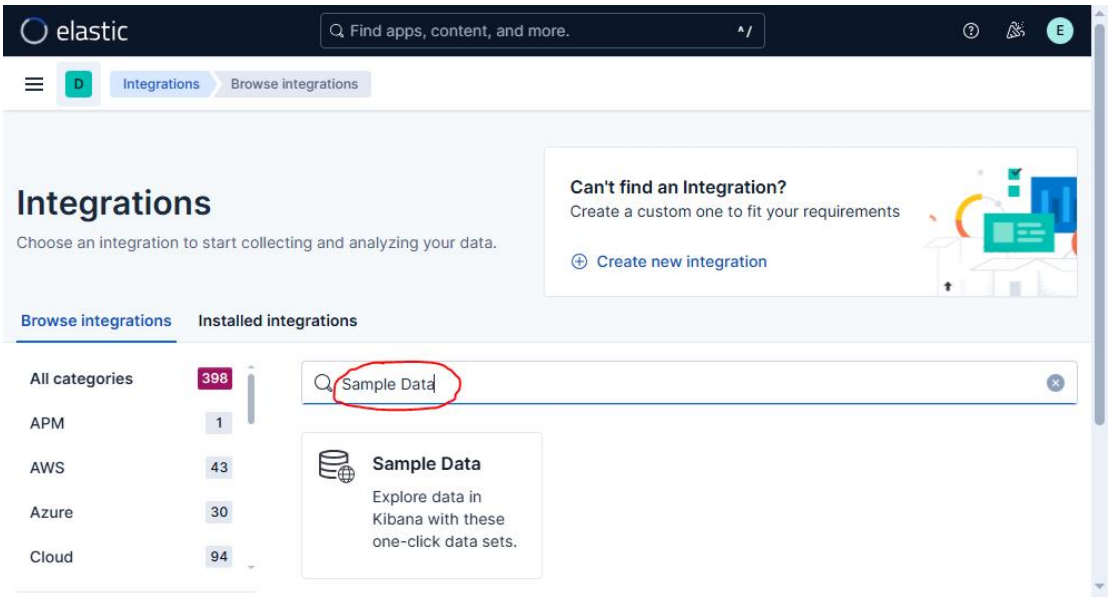
2.7.2. Перейдіть за адресою: http://localhost:5601.

2.7.3. Ви повинні побачити інтерфейс Kibana. Оскільки ви налаштували безпеку в Elasticsearch, Kibana запросить у вас облікові дані для входу (зазвичай це elastic та пароль, який ви встановили під час налаштування безпеки Elasticsearch, або облікові дані користувача kibana_system).

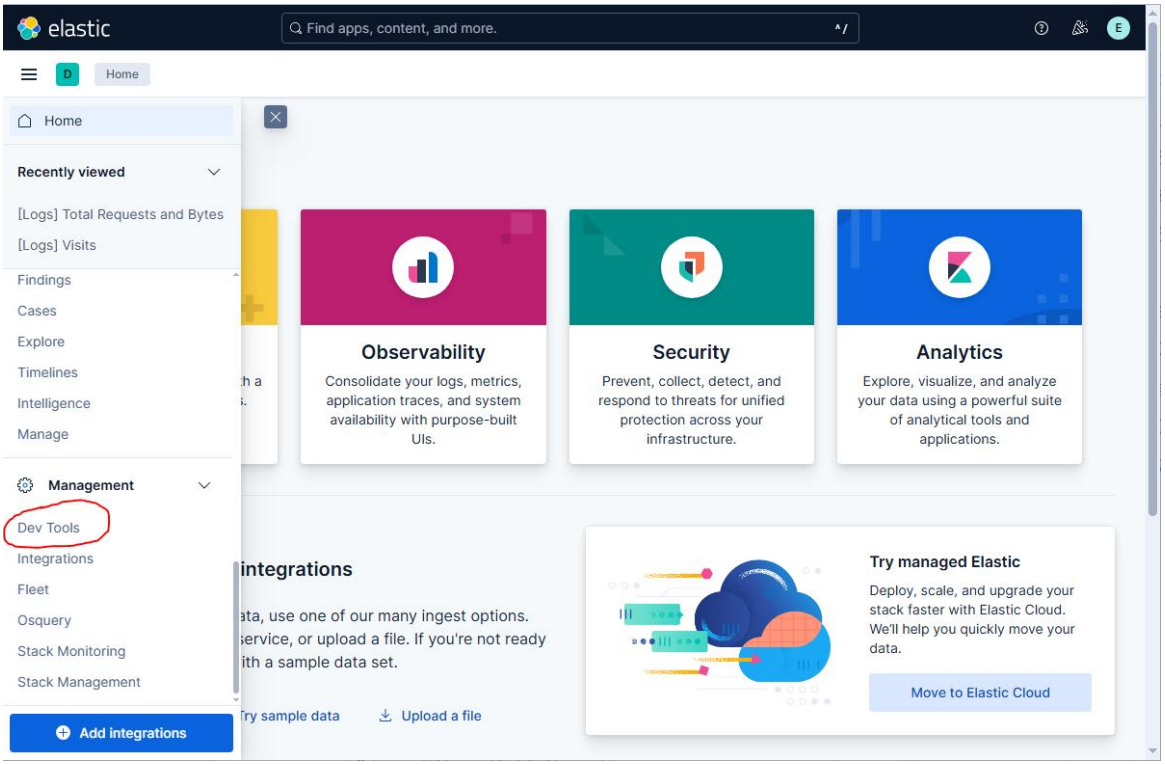


2.8. Для візуалізації логів та навчання, Kibana пропонує **вбудовані набори прикладів даних**. Ви можете знайти їх на головній сторінці Kibana або через розділ **"Integrations"** → **"Sample Data"**. Встановіть один з них (наприклад, "Sample Web Logs"), щоб побачити, як виглядає візуалізація даних у Kibana.

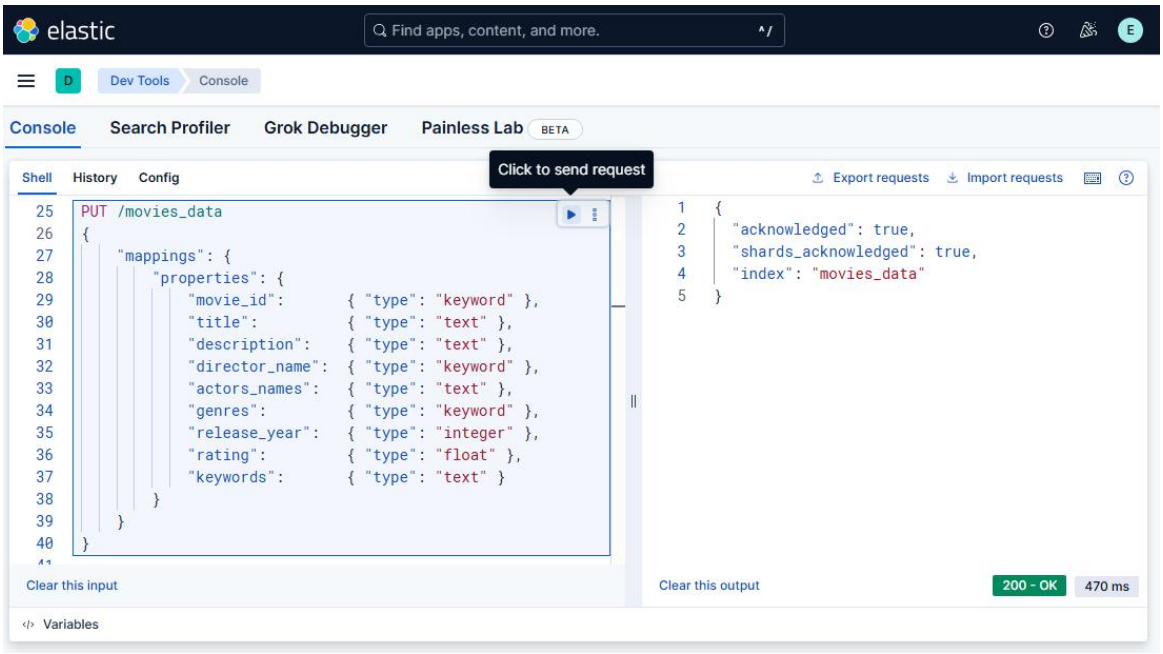


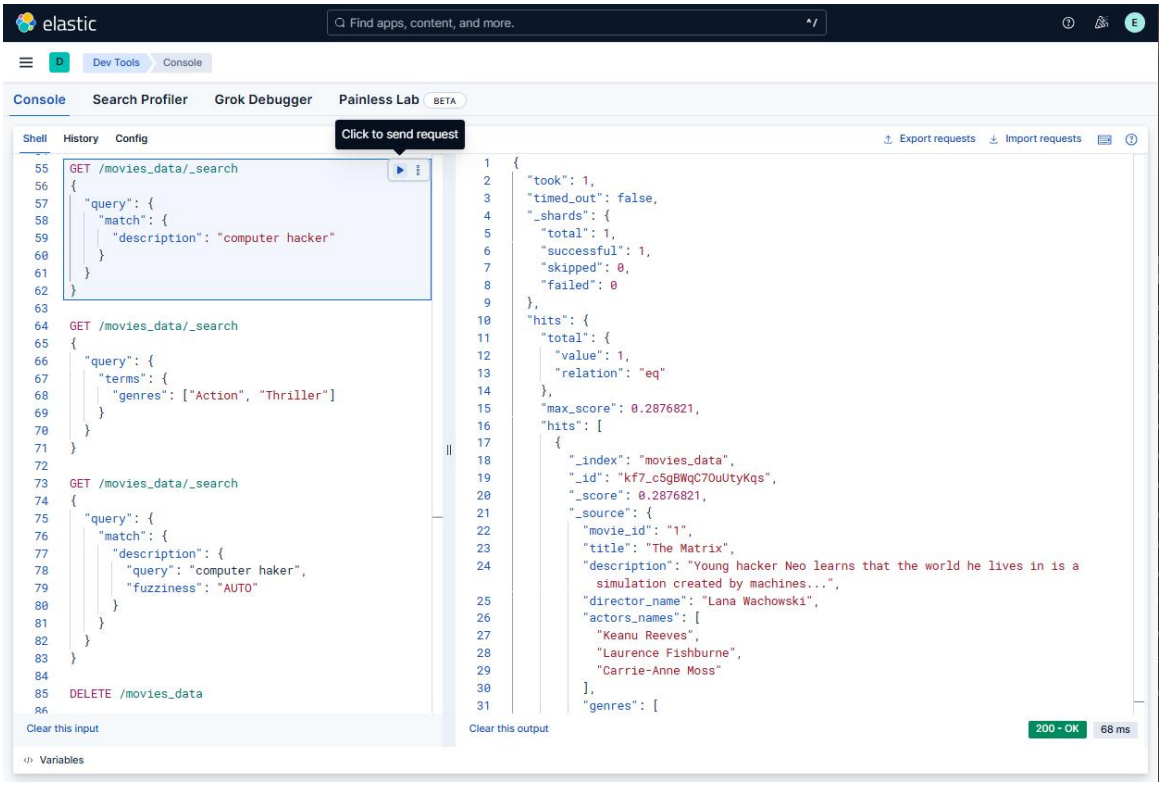


3. Відкрийте панель Dev Tools у Kibana з секції Management. Цей інструмент дозволяє писати запити до даних, що зберігаються у Elasticsearch. Використовуйте ліву панель для написання запитів. В правій панелі ви будете бачити результат виконання запиту.



3.1. У якості тренування повторіть ті запити, які було розглянуто на лекції.





3.2. Для закріплення навичок додайте до індексу декілька фільмів.

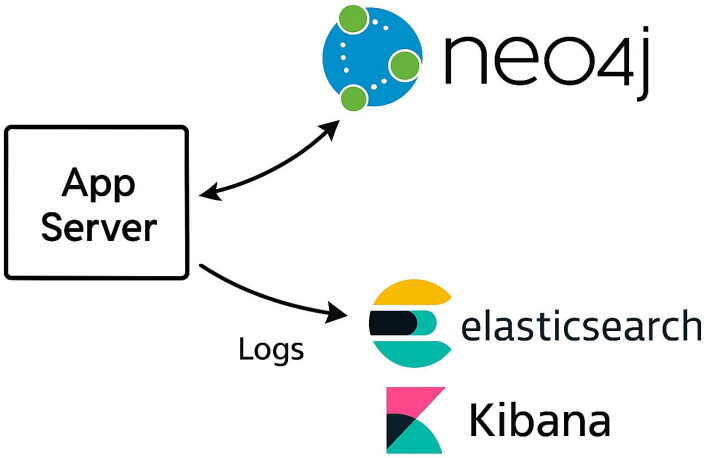
3.3. Напишіть пошукові запити, які дозволять:

- знайти фільми, в яких знімався певний актор;
- знайти фільми за певним описом, наприклад, “Азія нетрі мільйонер” (чи буд-який інший, в залежності від того, які фільми ви додали до індексу);
- знайти фільми за описом (або за назвою чи актором), навмисно припустившись кількох синтаксичних помилок в словах (використавши для для цього "fuzziness": "AUTO").

ЧАСТИНА 2 (оцінюється у **5 балів**).

4. До застосунку, розробленого в попередній лабораторній роботі (що була присвячена графовим СУБД) додати логування роботи застосунку. У випадку, якщо було розроблено клієнт-серверний засосунок з багатопотоковим сервером, логування дій потрібно здійснювати саме на сервері. Для логування рекомендується використовувати відповідні бібліотеки. Базовий приклад використання python-бібліотеки logging наведено в Додатку А.

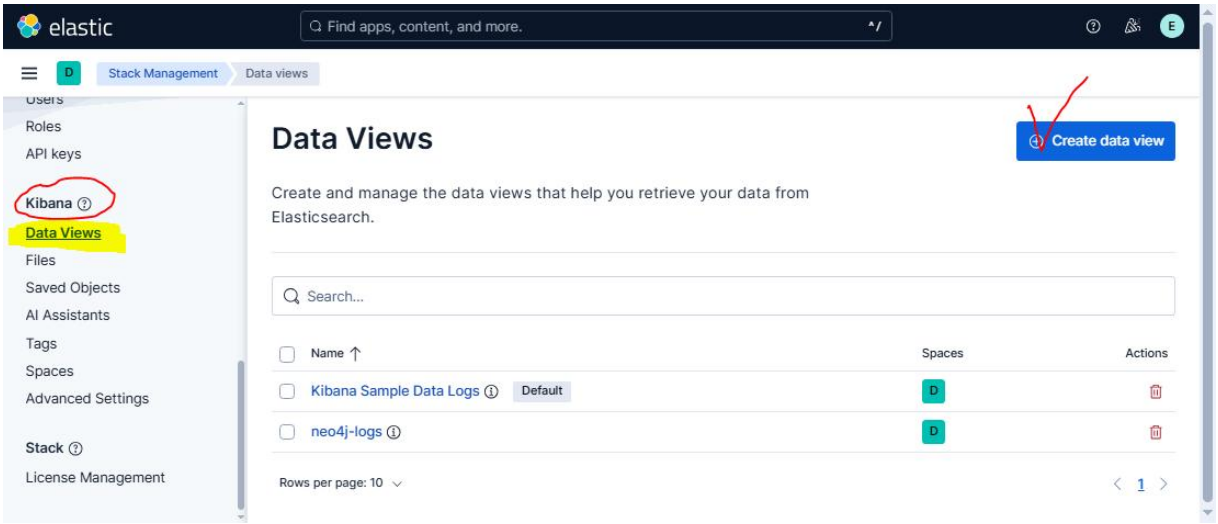
5. Вивід логів здійснювати в СУБД Elasticsearch.



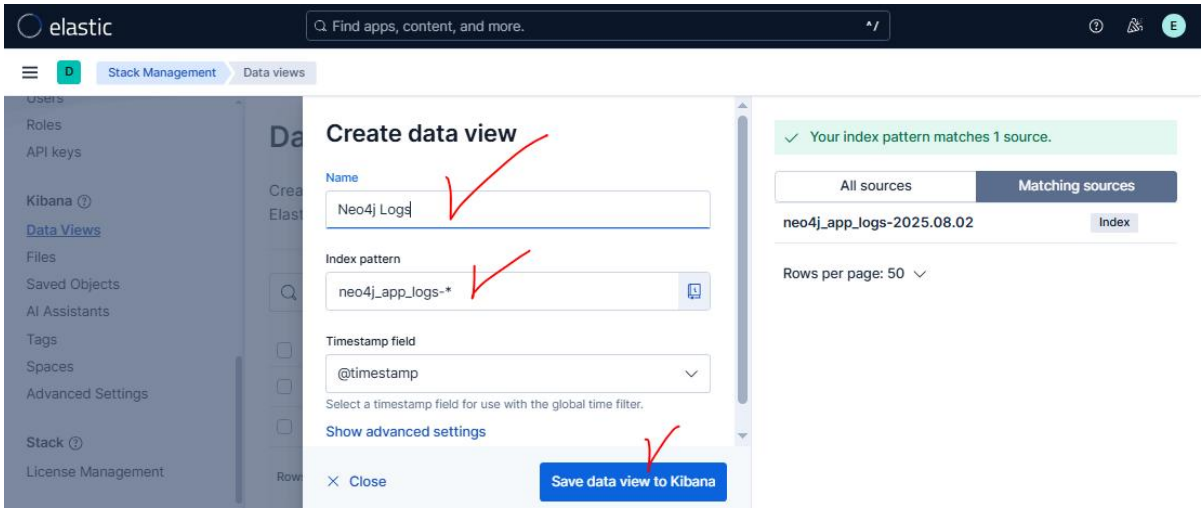
Для взаємодії з СУБД використовувати бібліотеку, специфічну для мови програмування, на якій розроблено сам застосунок. Приклад запису логів до Elasticsearch з використанням python-бібліотеки elasticsearch наведено в Додатку В (спрощений приклад також можна знайти в лекційній презентації). Перевірити наявність збережених в Elasticsearch логів за допомогою запитів в Kibana (скориставшись інструментом Dev Tools в Management)

6. За допомогою Kibana проаналізувати логи з використанням функціоналу Discover Visualize Library з блоку Analytics. Зокрема, побудувати діаграму, що ілюструє кількість логів кожного з рівнів: INFO, WARNING, ERROR, CRITICAL. Також побудувати діаграму, що ілюструє розподіл логів рівня ERROR по днях (або по годинах, якщо тестування відбувалось протягом одного дня). За допомогою Discover зNeo4j знайти логи, які стосувались помилок аутентифікацій при підключенні до СУБД. Також подивитись на їх розподіл за часом.

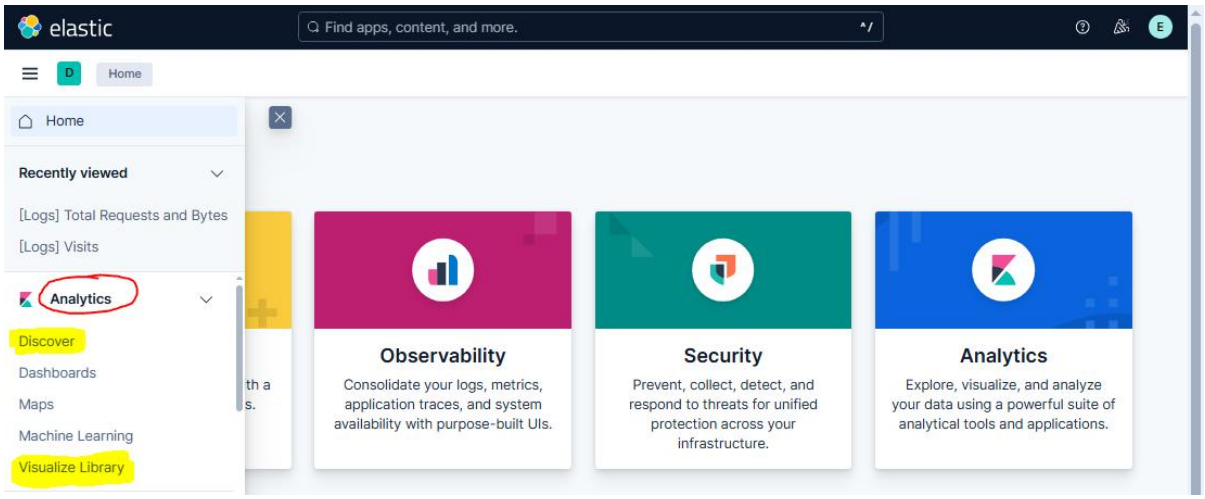
6.1. Перш ніж робити аналітику, треба налаштувати в Kibana патерн даних, з яким ви будете працювати. Це можна зробити за допомогою інструменту Data Views (Managemtnt → Stack Management → Kibana → Data Views).



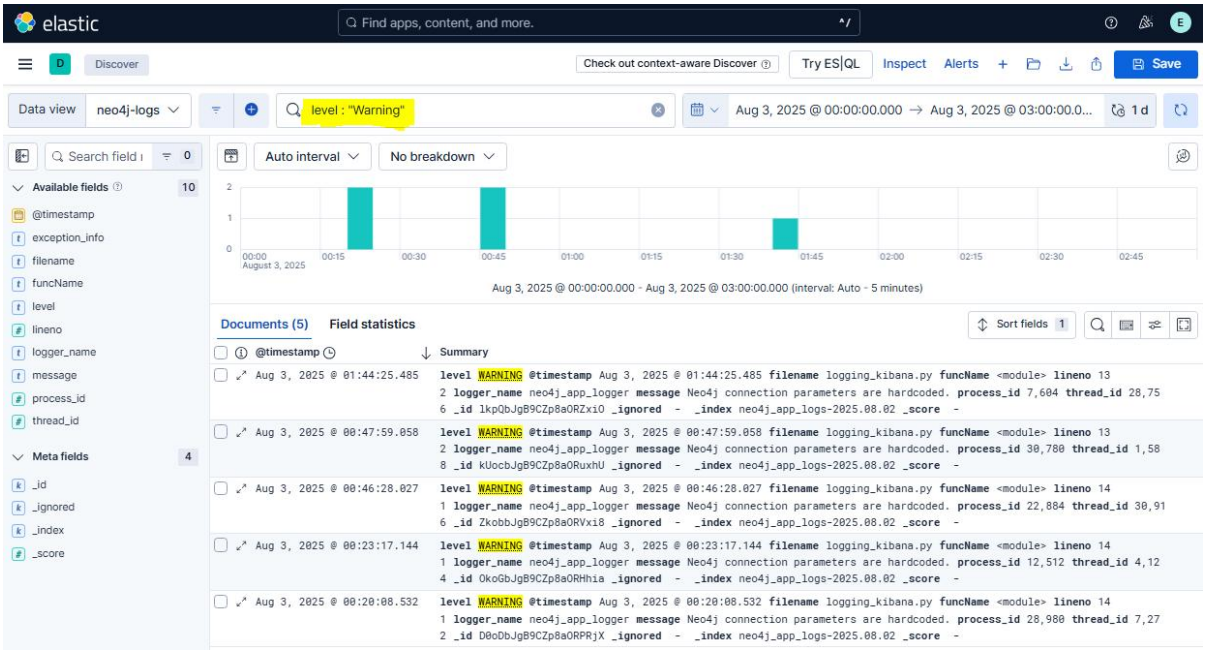
Треба задати патерн для тих індексів, в які ваш застосунок зберігав логи:



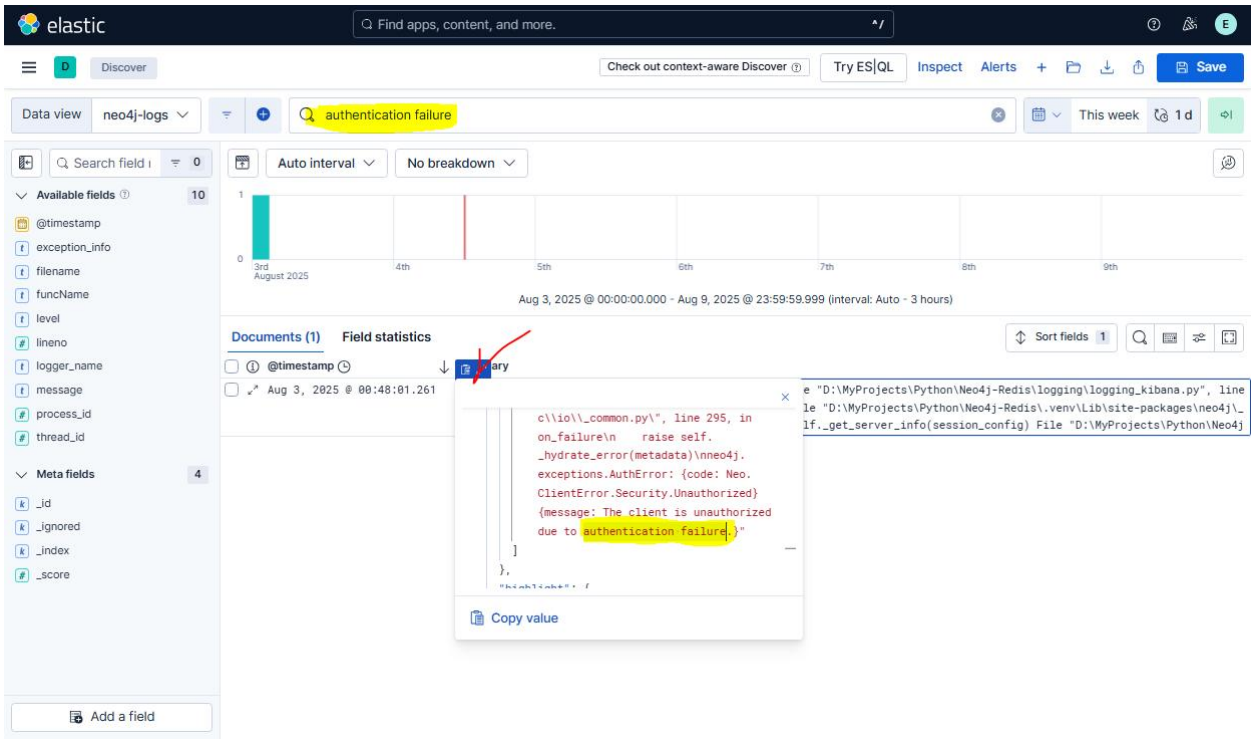
6.2. Використовуючи функціонал Analytics → Discover здійснюєте фільтруваня даних за рівнем логів, датою, часом, ключовими словами у текстах логів (наприклад, «authentication failure»).



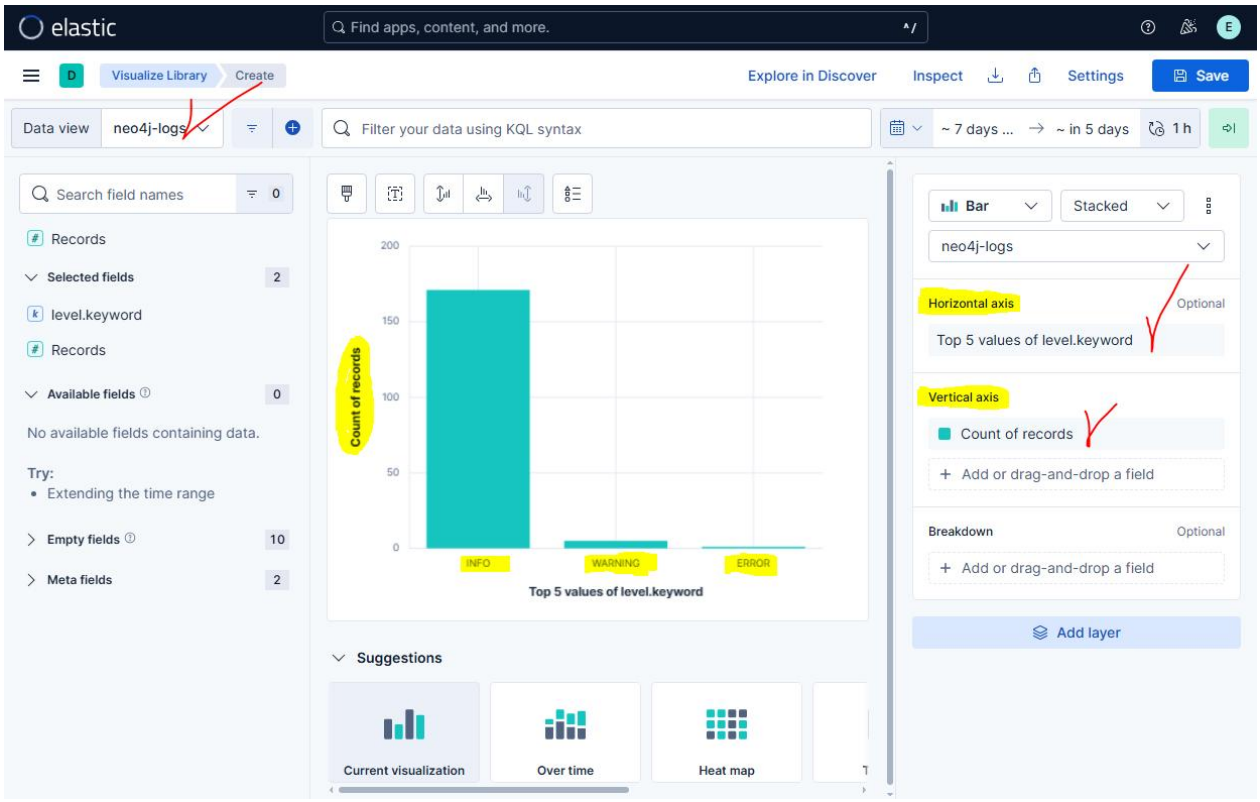
Цей інструмент (Discover) також дозволяє бачити розподіл відфільтрованої категорії логів за часом:



В тексті збережених повідомлень можна побачити весь стек викликів, який спричинив певну помилку.



6.3. Використовуючи функціоналу Analytics → Visualize Library → Create Visualization → Lens можна побудувати різні графіки та діаграми. Необхідно буде обрати тип графіку/діаграми та поля, поля, які ви хочете побачити на діаграмі та потрібну вам статистику (кількість, середнє значення тощо).



Примітка! Перелік діаграм та аналітичних запитів, які треба виконати в цьому пункті, не є жорстким і може бути скорегований за бажанням.

7. Оформити звіт, до якого додати скріншоти, що демонструють виконання всіх етапів завдання. До звіту включити також відповіді на контрольні запитання.

КОНТРОЛЬНІ ЗАПИТАННЯ

1. До якого типу NoSQL СУБД відноситься Elasticsearch? Які ще СУБД з цього класу є популярними на поточний момент? Аргументуйте свою відповідь, прикріпивши посилання на відповідні рейтинги.
2. Які сценарії використання пошукових СУБД ви знаєте?
3. Під який тип запитів в першу чергу оптимізовані пошукові СУБД?
4. У якому вигляді зберігає дані СУБД Elasticsearch?
5. Чи підтримує Elasticsearch механізм реплікацій та шардингу?

Додаток А.

Приклад логування на Python:

```
import logging

LOG_FILE_NAME = 'app_activity.log'

# -- 1. Створення та налаштування логгера ---
app_logger = logging.getLogger('my_app_logger')
# Встановлюємо мінімальний рівень для логгера на DEBUG.
app_logger.setLevel(logging.DEBUG)

# --- 2. Створення та налаштування обробника логів для консолі (StreamHandler)
console_handler = logging.StreamHandler()
# Встановлюємо рівень для консолі на DEBUG:
# в консоль будуть записуватися тільки повідомлення рівня DEBUG і вище.
console_handler.setLevel(logging.DEBUG)

# --- 3. Створення та налаштування обробника логів для файлу (FileHandler) ---
file_handler = logging.FileHandler(LOG_FILE_NAME, mode='w', encoding='utf-
8')
# Встановлюємо рівень для файлу на INFO:
# У файл будуть записуватися тільки повідомлення рівня INFO і вище.
file_handler.setLevel(logging.INFO)

# --- 4. Налаштування форматування (Formatter) ---
# Визначаємо формат лог-повідомлень.
# Можна використовувати один і той же форматер для різних обробників,
# або створити окремі, якщо потрібне різне форматування.
# %(asctime)s: час події
# %(name)s: ім'я логгера
```

```
# %(levelname)s: рівень логу (INFO, ERROR тощо)
# %(message)s: саме повідомлення логу
formatter = logging.Formatter('%(asctime)s - %(name)s - %(levelname)s -
%(message)s')

# Застосовуємо форматер до обох обробників.
console_handler.setFormatter(formatter)
file_handler.setFormatter(formatter)

# --- 5. Додавання обробників до логгера ---
app_logger.addHandler(console_handler)
app_logger.addHandler(file_handler)

# -- 6. Демонстрація логування на різних рівнях --
# Повідомлення рівня DEBUG (буде виведено тільки в консоль)
app_logger.debug("This is a DEBUG message. You will see it only in the
console.")

# Повідомлення рівня INFO (буде виведено і в консоль і в файл)
app_logger.info("Application started successfully.")

# Повідомлення рівня WARNING (буде виведено і в консоль і в файл)
app_logger.warning("Configuration file not found. Using default settings.")

# --- 7. Демонстрація логування виключення (ділення на нуль) ---
try:
    numerator = 10
    denominator = 0
    app_logger.info(f"Attempting to divide {numerator} by {denominator}...")
    result = numerator / denominator
except ZeroDivisionError as e:
    # Логуємо помилку рівня ERROR.
    # exc_info=True додає до логу трасування стека викликів (stack traceback).
    # Це повідомлення буде виведено і в консоль, і в файл.
    app_logger.error(f"A division by zero error occurred: {e}",
exc_info=True)
except Exception as e:
    # Загальний обробник для інших несподіваних виключень
    app_logger.critical(f"An unexpected critical error occurred: {e}",
exc_info=True)

# Повідомлення рівня CRITICAL (буде виведено і в консоль, і в файл).
# Виключно для демонстрації
app_logger.critical("Critical system failure detected. Shutting down.")
```

Додаток В.

Приклад запису логів до Elasticsearch:

```
import neo4j
import logging
from elasticsearch import Elasticsearch, ConnectionError, AuthenticationException,
```

```
TransportError
from urllib3.exceptions import ClosedPoolError
import datetime
import sys

# --- 1. Конфігурація Neo4j ---
DATABASE_NAME = "neo4j"
URI = "bolt://localhost:7687"
USERNAME = "neo4j"
PASSWORD = "01092025"

# --- 2. Конфігурація Elasticsearch ---
ELASTICSEARCH_HOSTS = ["http://localhost:9200"]
ELASTICSEARCH_USERNAME = "elastic"
ELASTICSEARCH_PASSWORD = "q+4mb0vDv6tvXV+o02im"

# Ім'я індексу, куди будуть записуватися логи.
# Рекомендується використовувати індекси на основі дати (наприклад, 'neo4j_app_logs-
YYYY.MM.DD').
LOG_INDEX_PREFIX = "neo4j_app_logs-"

# --- 3. Створення кастомного обробника для Elasticsearch ---
class ElasticsearchHandler(logging.Handler):
    """
    Кастомний обробник логів, який відправляє логи до Elasticsearch.
    Покладається на автоматичне створення індексу Elasticsearch при першому логуванні за
день.
    """
    def __init__(self, es_client, index_prefix):
        super().__init__()
        self.es_client = es_client
        self.index_prefix = index_prefix

    def emit(self, record):
        """
        Відправляє запис логу до Elasticsearch.
        Elasticsearch автоматично створить індекс, якщо його ще немає.
        """
        try:
            # Обчислюємо ім'я індексу для поточного запису на основі поточної дати
            index_name = self.index_prefix + datetime.datetime.now().strftime("%Y.%m.%d")

            # Форматуємо запис логу у словник, придатний для Elasticsearch
            log_entry = self.format_record_to_dict(record)

            # Індексуємо документ у відповідний щоденний індекс.
            # Elasticsearch автоматично створить індекс, якщо він не існує.
            self.es_client.index(index=index_name, document=log_entry)

        except (ConnectionError, ClosedPoolError) as ce:
            # Логуємо помилки з'єднання або закриття пулу безпосередньо в stderr,
            # щоб уникнути рекурсії.
            print(f"ERROR in ElasticsearchHandler: Could not connect to Elasticsearch or
pool is closed during log indexing. Please ensure it is running and accessible: {ce}",
file=sys.stderr)
        except AuthenticationException as ae:
            print(f"ERROR in ElasticsearchHandler: Authentication failed for Elasticsearch
during log indexing. Check your username and password: {ae}", file=sys.stderr)
        except TransportError as te:
            # Обробка помилок, пов'язаних з транспортом Elasticsearch (наприклад, помилки
індексації)
```

```
            print(f"ERROR in ElasticsearchHandler: Elasticsearch transport error during
log indexing: {te.info}", file=sys.stderr)
        except Exception as e:
            # Загальна обробка непередбачених помилок
            print(f"ERROR in ElasticsearchHandler: An unexpected error occurred while
sending log to Elasticsearch: {e}", file=sys.stderr)
        finally:
            self.handling_error = False # Скидаємо прапорець після завершення обробки

    def format_record_to_dict(self, record):
        """
        Перетворює об'єкт LogRecord на словник, який буде індексуватися в Elasticsearch.
        Включає стандартні атрибути логу та інформацію про винятки.
        """
        # Використовуємо стандартний форматер обробника
        message = self.format(record)
        exc_info_text = None
        if record.exc_info:
            exc_info_text = self.formatter.formatException(record.exc_info)

        log_dict = {
            # Використовуємо ISO-формат для дати, що є стандартном для Elasticsearch
            "@timestamp": datetime.datetime.fromtimestamp(record.created).isoformat(),
            "level": record.levelname,
            "logger_name": record.name,
            "message": message,
            "filename": record.filename,
            "lineno": record.lineno,
            "funcName": record.funcName,
            "process_id": record.process,
            "thread_id": record.thread,
        }

        if exc_info_text:
            log_dict["exception_info"] = exc_info_text

        return log_dict

# --- 4. Ініціалізація Elasticsearch клієнта та налаштування логгера ---
# Ініціалізуємо клієнт Elasticsearch глобально, щоб він був доступний у функції
connect_neo4j
es_client = None
# Налаштовуємо логгер додатка.
neo4j_logger = logging.getLogger('neo4j_app_logger')
neo4j_logger.setLevel(logging.DEBUG)
neo4j_logger.propagate = False # Важливо: відключаємо розповсюдження логів до кореневого
логгера

try:
    # Створення екземпляра клієнта Elasticsearch
    es_client = Elasticsearch(
        ELASTICSEARCH_HOSTS,
        # Використання базової автентифікації, якщо вказані ім'я користувача та пароль
        basic_auth=(ELASTICSEARCH_USERNAME, ELASTICSEARCH_PASSWORD) if
ELASTICSEARCH_USERNAME else None,
    )
    # Перевірка з'єднання з Elasticsearch. Якщо не вдасться, буде викинуто виняток.
    es_client.info()
    neo4j_logger.info("Successfully connected to Elasticsearch.")
    # Створюємо екземпляр нашого кастомного обробника Elasticsearch
    es_handler = ElasticsearchHandler(es_client, LOG_INDEX_PREFIX)
```

```
    es_handler.setLevel(logging.INFO) # Обробник буде відправляти в ES тільки INFO і вище
    # Форматуємо повідомлення, яке буде відправлено в Elasticsearch (тільки саме
повідомлення)
    formatter = logging.Formatter('%(message)s')
    es_handler.setFormatter(formatter) # Це встановлює self.formatter в обробнику
    # Додаємо наш обробник до логгера
    neo4j_logger.addHandler(es_handler)

    # --- 5. Логування тестових повідомлень (для перевірки) ---
    neo4j_logger.debug("This is a DEBUG message. It will NOT be sent to Elasticsearch
because handler level is INFO.")
    neo4j_logger.info("Application started. Connecting to Neo4j.")
    neo4j_logger.warning("Neo4j connection parameters are hardcoded.")

except ConnectionError as e:
    # Якщо не вдалося підключитися до ES, логуємо критичну помилку та завершуємо програму.
    # Використовуємо print, оскільки логгер ще не повністю налаштований для ES.
    print(f"CRITICAL ERROR: Failed to connect to Elasticsearch. Please ensure it's running
and accessible: {e}", file=sys.stderr)
    sys.exit(1) # Завершити програму, якщо підключення до ES не працює
except AuthenticationException as e:
    print(f"CRITICAL ERROR: Authentication failed for Elasticsearch. Check your
username/password: {e}", file=sys.stderr)
    sys.exit(1)
except Exception as e:
    print(f"CRITICAL ERROR: An unexpected error occurred during Elasticsearch client
initialization: {e}", file=sys.stderr)
    sys.exit(1)

# --- 6. Функція для підключення до Neo4j та виконання запиту ---
def connect_neo4j():
    global driver # Використовуємо глобальну змінну driver
    driver = None # Ініціалізуємо driver як None на початку функції
    try:
        neo4j_logger.info(f"Attempting to establish connection to Neo4j at {URI}.")
        driver = neo4j.GraphDatabase.driver(URI, auth=(USERNAME, PASSWORD))

        # Явна перевірка підключення до Neo4j
        driver.verify_connectivity()

        with driver.session(database=DATABASE_NAME) as session:
            neo4j_logger.info(f"The connection to Neo4j opened successfully.")

            # Коригуємо запит, щоб повертати 'item' для доступу до record["item"]["title"]
            query = "MATCH (item:Movie) RETURN item"
            neo4j_logger.debug(f"Executing Cypher query: {query}")

            result = session.run(query)
            neo4j_logger.debug("Querying data from Neo4j completed.")

            # Проходимо по результатах та логуємо заголовки фільмів
            for record in result:
                try:
                    movie_title = record["item"].get("title", "N/A (Title Missing)")
                    print(f"Movie title: {movie_title}")
                    neo4j_logger.info(f"Processed movie: {movie_title}")
                except KeyError as ke:
                    # Логуємо попередження, якщо очікуване поле відсутнє в записі
                    neo4j_logger.warning(f"KeyError: Missing expected 'title' key in movie
record. {ke}. Record data: {record.data()}", exc_info=False)
                except Exception as inner_e:
```

```
                    # Логуємо інші непередбачені помилки під час обробки запису
                    neo4j_logger.error(f"An unexpected error occurred while processing a
movie record: {inner_e}", exc_info=True)

    except neo4j.exceptions.ServiceUnavailable as e:
        # Логуємо помилки, коли Neo4j не запущений або недоступний
        neo4j_logger.error(f"Neo4j connection failed: Service is unavailable. Please
ensure Neo4j is running at {URI}. {e}", exc_info=True)
    except neo4j.exceptions.AuthError as e:
        # Логуємо помилки автентифікації Neo4j
        neo4j_logger.error(f"Neo4j authentication failed: Incorrect credentials for
{USERNAME}. {e}", exc_info=True)
    except neo4j.exceptions.ClientError as ce:
        # Логуємо помилки Cypher-запиту (синтаксис, семантика)
        neo4j_logger.error(f"Neo4j client error during query execution: {ce}",
exc_info=True)
    except Exception as e:
        # Логуємо будь-які інші непередбачені винятки
        neo4j_logger.critical(f"An unhandled critical exception occurred during Neo4j
operation: {e}", exc_info=True)
    finally:
        # Завжди закриваємо драйвер Neo4j, якщо він був ініціалізований
        if driver:
            driver.close()
            neo4j_logger.info(f"The connection to Neo4j closed successfully.")

if __name__ == "__main__":
    connect_neo4j()
```