Project 05

1. Title: Data Collection & Storage Foundation
2. Thời gian
    - Ngày bắt đầu: 3/3/2025
    - Ngày hoàn thành: 17/03/2025
3. Công cụ sử dụng GCP, Python, MongoDB, Virtual Machine, pymongo, IP2Location

2) GCP setup

Create Google Cloud Storage bucket

To create a bucket:

1. In the Cloud console, go to **Navigation menu ()** > **Cloud Storage** > **Buckets**.
2. Click **+ Create**:

- Name: bucket_project05
- Choose **Region** for **Location type** and `us-east1` for **Location**.
- Choose **Standard** for **default storage class**.

- Choose **Uniform** for **Access control** and **uncheck** *Enforce public access prevention on this bucket* to turn it off.

So sánh các lớp lưu trữ

Lớp lưu trữ	Tần suất truy cập	Chi phí lưu trữ	Chi phí truy xuất	Thời gian lưu trữ tối thiểu
Standard	Thường xuyên	Cao	Không có	Không có
Nearline	1 lần/tháng	Thấp hơn Standard	Có	30 ngày
Coldline	1 lần/năm	Thấp hơn Nearline	Có	90 ngày
Archive	Dưới 1 lần/năm	Rất thấp	Cao	365 ngày

Setup Authentication

- Bước 1: Tạo Service Account
  - Điều hướng đến **IAM & Admin** → **Service Accounts**.
  - Chọn **Create Service Account** và nhập tên.
  - Gán quyền (ví dụ: `roles/editor`, `roles/storage.admin` tùy theo nhu cầu).
- Bước 2: Tạo Key cho Service Account
  - Chọn Service Account vừa tạo.
  - Chuyển đến tab **Keys** → **Add Key** → **Create new key**.
  - Chọn định dạng **JSON** và tải file xuống.
  - Lưu trữ file này an toàn, tránh bị lộ.

Thiết lập biến môi trường trên Linux

```
export GOOGLE_APPLICATION_CREDENTIALS="/path/to/your-key.json"
```

Kiểm tra Authentication

- Sử dụng gcloud CLI:
  Cài đặt gcloud

```
sudo apt update && sudo apt install google-cloud-sdk -y
```

- Kiểm tra xác thực

```
gcloud auth activate-service-account --key-file=$GOOGLE_APPLICATION_CREDEI
```

- Test quyền bằng cách liệt kê các project:

```
gcloud projects list
```

- Kiểm tra tài khoản đang sử dụng

```
gcloud auth list
```

Bước cuối: Sử dụng Service Account trong ứng dụng

Ví dụ 1: Python sử dụng Service Account

Test file uploads

3) Setup virtual machine

Setup VM instance

1. In the **Cloud console**, on the **Navigation menu** (≡), click **Compute Engine** > **VM Instances**.

   > This may take a minute to initialize for the first time.

2. To create a new instance, click **Create Instance**.
3. In the **Machine configuration**:

5. Click **OS and storage**.

   Click **Change** to begin configuring your boot disk and select the values for:

- **Operating system**: Debian
- **Version**: Debian GNU/Linux 11 (bullseye)
- **Boot disk type**: Balanced persistent disk
- **Size**: 10 GB

Click **Networking**.

6. **Firewall**: Allow HTTP traffic

Install MongoDB on the VM

Cài Đặt `mongosh` (MongoDB Shell) trên Debian 11

1. Thêm kho lưu trữ MongoDB:

```
wget -qO - https://www.mongodb.org/static/pgp/server-7.0.asc | sudo apt-key add -
echo "deb [ arch=amd64,arm64 ] https://repo.mongodb.org/apt/ubuntu $(lsb_release
sudo apt-get update
```

2. Cài đặt MongoDB:

```
sudo apt-get install -y mongodb-org
```

3. Khởi động MongoDB:

```
sudo systemctl start mongod
sudo systemctl enable mongod
```

4. Kiểm tra `mongosh` đã hoạt động chưa

```
mongosh
```

Lưu ý

- MongoDB của không khởi động được do lỗi **"exit-code 14"**, thường liên quan đến quyền truy cập, thiếu thư mục dữ liệu hoặc file cấu hình sai.

Cách xử lý

Kiểm tra nhật ký lỗi

```
sudo cat /var/log/mongodb/mongod.log
```

Lỗi chính trong log là:

`"Failed to unlink socket file" → "Operation not permitted"`

MongoDB không thể xóa file socket `/tmp/mongodb-27017.sock`, dẫn đến lỗi khởi động.

→

1. **Xóa file socket lỗi**

```
sudo rm -f /tmp/mongodb-27017.sock
```

2. **Kiểm tra quyền thư mục** `/tmp`

Đảm bảo thư mục `/tmp` có quyền phù hợp:

3. **Xác minh quyền MongoDB**

Chắc chắn rằng MongoDB có quyền trên thư mục dữ liệu:

```
sudo chown -R mongodb:mongodb /var/lib/mongodb /var/log/mongodb
sudo chmod -R 755 /var/lib/mongodb /var/log/mongodb
```

4. Khởi động lại MongoDB

```
sudo systemctl restart mongod
sudo systemctl status mongod
```

Test connectivity

1️⃣ Kiểm tra MongoDB có đang chạy không

```
ps aux | grep mongod
```

Kiểm tra kết nối với MongoDB bằng `mongosh`

```
mongosh
```

Nếu thành công, sẽ thấy:

```
Current Mongosh Log ID: 65a2c3...
Connecting to: mongodb://127.0.0.1:27017
```

Sau đó, thử chạy:

```
show dbs
```

Error!

```
mongod --dbpath /data/db --logpath /var/log/mongodb.log
{"t":{"$date":"2025-03-03T15:17:09.389Z"},"s":"F",  "c":"CONTROL",
 "id":20574,   "ctx":"-","msg":"Error during global initialization",
 "attr":{"error":{"code":37,"codeName":"FileRenameFailed","errmsg":
 "Could not rename preexisting log file \"/var/log/mongodb.log\" to
 \"/var/log/mongodb.log.2025-03-03T15-17-09\"; run with --logappend or
  manually remove file: Permission denied"}}}
```

Lỗi này xảy ra do **MongoDB không có quyền ghi hoặc đổi tên file log** trong thư mục `/var/log/`.

Cách xử lý:

- **Cấp quyền cho file log, Xóa file log cũ (nếu có):**

```
sudo rm -f /var/log/mongodb.log
```

- Tạo file log mới và cấp quyền:

```
sudo touch /var/log/mongodb.log
sudo chown `whoami`:`whoami` /var/log/mongodb.log
sudo chmod 644 /var/log/mongodb.log
```

- Chạy lại MongoDB:

```
mongod --dbpath /data/db --logpath /var/log/mongodb.log --fork
```

Kết nối tới VM từ local

Cài đặt project đang chạy là Project-5 với project id là project-452300

```
gcloud config set project project-452300
```

Tạo SSH Key Mới Trên Máy Local

```
ssh-keygen -t rsa -b 4096 -f ~/.ssh/my_new_gcp_key -C "hofonam"
```

Thêm Key Mới Vào Metadata của VM

```
gcloud compute instances add-metadata project-05-vm \
   --metadata=ssh-keys="hofonam:$(cat ~/.ssh/my_new_gcp_key.pub)" \
   --zone=us-central1-a --project=project-452300
```

Kết Nối Lại Bằng SSH

```
ssh -i ~/.ssh/my_new_gcp_key hofonam@35.192.69.127
```

4) Initial Data Loading

- Import raw data into MongoDB

Vì đã upload file `glamira_ubl_oct2019_nov2019.tar.gz` lên buckets gcp, nên ở bước này, thì cần download file từ bucket về VM.

Error!

Vì chưa cấp quyền read access cho tài khoản về tải file về từ Buckets nên gặp lỗi Permission denied!

- Cách xử lý: Cấp quyền cho tài khoản đang sử dụng:

```
gcloud projects add-iam-policy-binding project-452300 \
  --member=serviceAccount:736075149505-compute@developer.gserviceaccount.co
  --role=roles/storage.objectViewer
```

- Download file từ Buckets về VM

```
gsutil cp gs://bucket_project05/glamira_ubl_oct2019_nov2019.tar.gz ~/
```

Error!

Đối với cách này phải trực tiếp extract file data sau đó mới có thể import vào MongoDB, vì vậy phải tốn hơn 30 GB bộ nhớ nên không thể làm trực tiếp được

→ Tạo 1 folder extracted_data/ trên Buckets của GCP sau đó extract ra folder đó và import vào MongoDB trên VM

Solution: Mount GCP bucket into VM and import data from bucket directly to MongoDB

1) Mount bucket to VM

1. Tải gcsfuse

```
export GCSFUSE_REPO=gcsfuse-`lsb_release -c -s`
echo "deb http://packages.cloud.google.com/apt $GCSFUSE_REPO main" | sudo tee /
curl https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key add -
sudo apt update
sudo apt install gcsfuse -y
```

2. Tạo 1 directory để mount bucket

```
mkdir ~/gcs_bucket
```

3. Mount bucket

```
gcsfuse bucket_project05 ~/gcs_bucket
```

Error: Với cách này khá khó vì dung lượng VM chỉ có 10GB và có dính tới các quyền truy cập trên Buckets

✈️ Nên mở rộng dung lượng VM và giải nén file

Mở rộng dung lượng VM

Bước 1: Cài đặt `sfdisk` , `growpart` và `cloud-guest-utils`

```
sudo apt update
sudo apt install fdisk cloud-guest-utils -y
sudo apt install cloud-guest-utils -y
```

Bước 2: Chạy lại lệnh mở rộng phân vùng

```
sudo growpart /dev/sda 1
sudo resize2fs /dev/sda1
```

2) Giải nén file data vào 1 folder tên là extracted_data/

```
# Tạo thư mục để chứa file giải nén
mkdir -p extracted_data

# Sao chép file từ GCS về VM
gsutil cp gs://bucket_project05/glamira_ubl_oct2019_nov2019.tar.gz .

# Giải nén file vào thư mục extracted_data
tar -xzvf glamira_ubl_oct2019_nov2019.tar.gz -C extracted_data/
```

Import raw data into MongoDB

```
mongorestore --db countly --collection summary
extracted_data/dump/countly/summary.bson
```

Thành phần	Giải thích
`mongorestore`	Công cụ khôi phục dữ liệu MongoDB từ file `.bson` .

`--db countly`	Chỉ định database đích là `countly`. Nếu database chưa tồn tại, MongoDB sẽ tự tạo.
`--collection summary`	Chỉ định collection đích là `summary`. Nếu collection chưa tồn tại, MongoDB sẽ tự tạo.
`extracted_data/dump/countly/summary.bson`	Đường dẫn đến file `.bson` chứa dữ liệu cần khôi phục.

Kiểm tra dữ liệu sau khi import

1. Mở Mongo shell:

```
mongosh --host 127.0.0.1 --port 27017
```

2. Chuyển sang database `countly`:

```
use countly
```

3. Kiểm tra dữ liệu:

```
db.summary.findOne()
```

4. Kiểm tra số lượng document đã import:

```
db.summary.countDocuments()
```

5. Kiểm tra indexes đã tạo:

```
db.summary.getIndexes()
```

Error!

Đang import data thì bị crash nguyên nhân do khi import đồng thời dung lượng data trên VM sẽ x2 dẫn đến không đủ dung lượng yêu cầu

→ Cách xử lý

```
mongorestore --db countly --collection summary --keepIndexVersion
extracted_data/dump/countly/summary.bson
```

`--keepIndexVersion` giúp giữ lại chỉ mục cũ mà không cần tạo lại, tránh lãng phí dung lượng.

Basic MongoDB queries to explore data

```
countly>db.summary.countDocuments()
41432473
```

→ hiện có **41,432,473 documents**.

```
countly>show dbs
admin     40.00 KiB
config   108.00 KiB
countly   12.11 GiB
local     72.00 KiB
```

đã có database countly sau khi import data vào MongoDB

```
countly> db.summary.getIndexes()
[
  { v: 2, key: { _id: 1 }, name: '_id_' },
  { v: 2, key: { device_id: 1 }, name: 'device_id_1' },
  { v: 2, key: { time_stamp: -1 }, name: 'time_stamp_-1' }
]
```

In ra tất cả các Indexes trong data này

Document data structure (collections, fields)

```
countly>show collections
summary
```

→ Có collection là summary

Lấy ra data mẫu

```
countly> db.summary.find().limit(1).pretty()
[
  {
    _id: ObjectId("5ed8cb2bc671fc36b74653ad"),
    time_stamp: 1591266092,
    ip: '37.170.17.183',
    user_agent: 'Mozilla/5.0 (iPhone; CPU iPhone OS 13_4_1 like Mac OS X) AppleWebKit
    resolution: '375×667',
    user_id_db: '502567',
    device_id: 'beb2cacb-20af-4f05-9c03-c98e54a1b71a',
    api_version: '1.0',
    store_id: '12',
    local_time: '2020-06-04 12:21:27',
    show_recommendation: 'false',
    current_url: 'https://www.glamira.fr/glamira-pendant-viktor.html?alloy=yellow-375',
    referrer_url: 'https://www.glamira.fr/men-s-necklaces/',
    email_address: 'pereira.vivien@yahoo.fr',
    recommendation: false,
    utm_source: false,
    utm_medium: false,
    collection: 'view_product_detail',
    product_id: '110474',
    option: [
      {
        option_label: 'alloy',
        option_id: '332084',
        value_label: '',
        value_id: '3279318'
      },
      {
        option_label: 'diamond',
```

```
      option_id: '',
      value_label: '',
      value_id: ''
    }
  ]
}
]
```

Create data dictionary

Field Name	Data Type	Description	Example Value
_id	ObjectId	Unique identifier for each document	ObjectId("5ed8cb2bc671fc36b74653ad")
time_stamp	Number (int)	Unix timestamp representing event time	1591266092
ip	String	User's IP address	"37.170.17.183"
user_agent	String	Browser & device information	"Mozilla/5.0 (iPhone; CPU iPhone OS 13_4_1... Safari/604.1")
resolution	String	Screen resolution of the device	"375×667"
user_id_db	String	User ID in the database	"502567"
device_id	String	Unique device identifier	"beb2cacb-20af-4f05-9c03-c98e54a1b71a"
api_version	String	API version used	"1.0"
store_id	String	ID of the store where the action occurred	"12"
local_time	String (Datetime)	Local time when the event happened	"2020-06-04 12:21:27"

show_recommendation	String (Boolean)	Whether recommendations were shown (true/false)	"false"
current_url	String (URL)	The URL of the page the user is currently viewing	"https://www.glamira.fr/glamira-pendant-viktor.html?alloy=yellow-375"
referrer_url	String (URL)	The previous URL that referred the user	"https://www.glamira.fr/men-s-necklaces/"
email_address	String (Email)	User's email address	"pereira.vivien@yahoo.fr"
recommendation	Boolean	Whether the user received a recommendation	false
utm_source	Boolean/String	Campaign source tracking (false if not available)	false
utm_medium	Boolean/String	Campaign medium tracking (false if not available)	false
collection	String	Type of user action	"view_product_detail"
product_id	String	ID of the product being viewed or interacted with	"110474"
option	Array of Objects	Product options selected by the user	[{ "option_label": "alloy", "option_id": "332084", "value_label": "", "value_id": "3279318" }]

5) IP Location Processing

Install ip2location-python library

Cập nhật danh sách gói của hệ thống:

```
sudo apt update && sudo apt upgrade -y
```

Cài đặt python và pip

```
sudo apt install python3 python3-pip -y
```

Cài đặt thư viện `ip2location`

```
pip install ip2location
```

Write Python script

1. Import pymongo

```
pip install pymongo
```

2. **Connect to MongoDB**

```
from pymongo import MongoClient, UpdateOne
import IP2Location
import logging

# Cấu hình logging để hiển thị ra cả file và màn hình
logging.basicConfig(
    filename="ip_location_processing.log",
    level=logging.INFO,
    format="%(asctime)s - %(levelname)s - %(message)s",
)
console_handler = logging.StreamHandler()
console_handler.setLevel(logging.INFO)
formatter = logging.Formatter("%(asctime)s - %(levelname)s - %(message)s")
console_handler.setFormatter(formatter)
logging.getLogger().addHandler(console_handler)


def process_ip_locations():
```

```
    """Xử lý dữ liệu từ MongoDB và cập nhật location cho từng IP, dùng bulk_write để tố
    try:
        # Kết nối MongoDB
        client = MongoClient("mongodb://localhost:27017/")
        db = client["countly"]
        old_collection = db["summary"]
        new_collection = db["summary_with_location"]

        # Khởi tạo thư viện IP2Location
        ip_db = IP2Location.IP2Location("IP2LOCATION-LITE-DB5.IPV6.BIN")

        batch_size = 1000  # Giảm batch size để tránh tốn RAM quá nhiều
        cursor = old_collection.find({"ip": {"$exists": True}})
        logging.info("Connected to MongoDB successfully!")

        bulk_operations = []
        total_processed = 0  # Biến đếm số lượng bản ghi đã xử lý

        for doc in cursor:
            ip_address = doc.get("ip")
            if ip_address:
                try:
                    record = ip_db.get_all(ip_address)
                    location_info = {
                        "ip": ip_address,
                        "country": record.country_long,
                        "region": record.region,
                        "city": record.city,
                        "latitude": record.latitude,
                        "longitude": record.longitude,
                    }
                    doc["location_info"] = location_info
                    logging.info(f"Processed IP: {ip_address} → {location_info['country']}, {loc
                except Exception as e:
                    logging.error(f"Error processing IP {ip_address}: {e}")
                    doc["location_info"] = None
            else:
                doc["location_info"] = None
```

```
            # Thêm vào danh sách bulk update
            bulk_operations.append(UpdateOne({"_id": doc["_id"]}, {"$set": doc}, upsert=T

            # Nếu đạt batch_size thì thực hiện bulk_write
            if len(bulk_operations) >= batch_size:
                new_collection.bulk_write(bulk_operations)
                total_processed += len(bulk_operations)
                logging.info(f"Inserted {total_processed} records into 'summary_with_locatio
                bulk_operations = []  # Reset batch

        # Xử lý batch còn lại
        if bulk_operations:
            new_collection.bulk_write(bulk_operations)
            total_processed += len(bulk_operations)
            logging.info(f"Inserted {total_processed} records into 'summary_with_location''

        logging.info("Hoàn thành cập nhật toàn bộ bản ghi.")

    except Exception as e:
        logging.error(f"Error in processing IP locations: {e}")


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

Up file code lên VM

```
scp -i ~/.ssh/my_new_gcp_key /home/hofonam/Documents/UniGap/Project05/project0
```

- Up file IP2LOCATION-LITE-DB5.IPV6.BIN lên VM để load cơ sở dữ liệu IP2Location

```
scp -i ~/.ssh/my_new_gcp_key /home/hofonam/Documents/UniGap/Project05/IP2LOC
```

Tạo collection mới để lưu trữ location

```
use countly
db.createCollection("summary_with_location")
```

Test with sample data

Đếm số lượng ip đến từ France

```
countly>db.summary_with_location.countDocuments({"location_info.country":"France
2186100
```

Đếm số lượng bản ghi có region là Nouvelle-Aquitaine

```
countly> db.summary_with_location.countDocuments({"location_info.region":"Nouvell
176221
```

Đếm số bản ghi theo từng quốc gia

```
countly> db.summary_with_location.aggregate([
...     { $group: { _id: "$location_info.country", count: { $sum: 1 } } },
...     { $sort: { count: -1 } }
... ]);

[
  { _id: 'United States of America', count: 5079071 },
  { _id: 'Germany', count: 4945700 },
  {
    _id: 'United Kingdom of Great Britain and Northern Ireland',
    count: 2896715
  },
  { _id: 'Sweden', count: 2197699 },
  { _id: 'France', count: 2186100 },
  { _id: 'Italy', count: 1642862 },
  { _id: 'Spain', count: 1517566 },
  { _id: 'Romania', count: 1250936 },
  { _id: 'Australia', count: 1236391 },
  { _id: 'Netherlands (Kingdom of the)', count: 1178036 },
```

```
  { _id: 'Mexico', count: 1159833 },
  { _id: 'Turkiye', count: 1145120 },
  { _id: 'Switzerland', count: 978212 },
  { _id: 'Canada', count: 966993 },
  { _id: 'Hungary', count: 909756 },
  { _id: 'Poland', count: 802712 },
  { _id: 'Belgium', count: 799983 },
  { _id: 'Chile', count: 781099 },
  { _id: 'Denmark', count: 730047 },
  { _id: 'Czechia', count: 729118 }
]
```

6) Product name collection

Filter data in collections equal to `view_product_detail`, `select_product_option`, and `select_product_option_quality`.

```
countly> db.summary_with_location.find({
...   collection: { $in: ["view_product_detail", "select_product_option", "select_product_
... }).pretty();
[
  {
    _id: ObjectId("5ed8cb2bc671fc36b74653ad"),
    api_version: '1.0',
    collection: 'view_product_detail',
    current_url: 'https://www.glamira.fr/glamira-pendant-viktor.html?alloy=yellow-375',
    device_id: 'beb2cacb-20af-4f05-9c03-c98e54a1b71a',
    email_address: 'pereira.vivien@yahoo.fr',
    ip: '37.170.17.183',
    local_time: '2020-06-04 12:21:27',
    location_info: {
      ip: '37.170.17.183',
      country: 'France',
      region: 'Nouvelle-Aquitaine',
      city: 'Sarbazan',
```

```
    latitude: '44.020290',
    longitude: '-0.312940'
   },
   option: [
    {
     option_label: 'alloy',
     option_id: '332084',
     value_label: '',
     value_id: '3279318'
    },
    {
     option_label: 'diamond',
     option_id: '',
     value_label: '',
     value_id: ''
    }
   ],
```

Get the `product_id` and `current_url` values.

```
countly> db.summary_with_location.find({},{product_id:1, current_url:1, _id : 0}).limit(10
[
 {
  current_url: 'https://www.glamira.fr/glamira-pendant-viktor.html?alloy=yellow-375',
  product_id: '110474'
 },
 {
  current_url: 'https://www.glamira.com.au/customcheckout/onepage/payment/'
 },
 {
  current_url: 'https://www.glamira.es/alianzas/?gclid=CjwKCAjwt-L2BRA_EiwAacX32
 },
 {
  current_url: 'https://www.glamira.com.au/glamira-diamonds-ohrstecker-louisa-skug
  product_id: '85796'
 },
 { current_url: 'https://www.glamira.bg/' },
 { current_url: 'https://www.glamira.sk/zasnubne-prstene/diamant/' },
```

```
  {
    current_url: 'https://www.glamira.com.au/sense-muse-4-mm.html?alloy=white_red-
    product_id: '99316'
  },
```

Crawl the product name based on the information above; get only one active `product name` for each distinct `product_id`

```
countly> db.summary_with_location.aggregate([
...   {
...     $match: {
...       collection: { $in: ["view_product_detail", "select_product_option", "select_produc
...     }
...   },
...   {
...     $group: {
...       _id: "$product_id",
...       current_url: { $first: "$current_url" }
...     }
...   },
...   {
...     $project: {
...       _id: 1,
...       product_name: {
...         $arrayElemAt: [{$split: [{$arrayElemAt: [{ $split: [{ $arrayElemAt: [{ $split: ["$cu
...       }
...     }
...   }
... ]).pretty();
[
  { _id: '101016', product_name: 'glamira-bracelet-delicias' },
  { _id: '92274', product_name: 'glamira-ring-amay' },
  { _id: '95662', product_name: 'glamira-pendant-topeng' },
  { _id: '98245', product_name: 'anillo-de-mujer-unique-joy-10-mm' },
  { _id: '91775', product_name: 'glamira-ring-serenity-10-mm' },
  { _id: '109087', product_name: 'womens-ring-magic-love-8-mm' },
  { _id: '108030', product_name: 'kadin-yuzuk-florid-glitz-5-mm' },
  { _id: '99492', product_name: 'anillo-de-hombre-infinite-wake-5-mm' },
```

```
  { _id: '106176', product_name: 'glamour-love-10-mm' },
  {
    _id: '105111',
    product_name: 'anillo-de-mujer-mysterious-touch-10-mm'
  },
  { _id: '90576', product_name: 'damenring-pretty-elegance' },
  { _id: '109686', product_name: 'glamira-ring-eladia' },
  { _id: '90944', product_name: 'glamira-ring-celine-0' },
  { _id: '111345', product_name: 'glamira-pendant-sibyl-l' },
  { _id: '102020', product_name: 'glamira-prsten-jeralee' },
  { _id: '111162', product_name: 'confident-glow-8-mm' },
  { _id: '110537', product_name: 'glamira-pendant-saturnina' },
  { _id: '92890', product_name: 'classic-tour-4mm' },
  { _id: '92615', product_name: 'anillo-de-hombre-classic-way' },
  { _id: '85172', product_name: 'womens-ring-smart-devotion-skup7014' }
```

Giải thích:

- B1: Chọn ra các sản phẩm thuộc 3 collection là:

  "view_product_detail", "select_product_option", "select_product_option_quality"

- B2: Group by product_id và current_url để lấy ra 1 URL mẫu dựa trên các product_id
- B3: Trích xuất product_name

URL mẫu sẽ có dạng như này:

https://www.glamira.fr/glamira-pendant-viktor.html?alloy=yellow-375

→ Cần trích xuất ra tên là “glamira-pendant-viktor” sử dụng phân tích chuỗi

- **$split: ["$current_url", "/"]** : Chia URL thành các phần ngăn cách bởi dấu “/” và lấy phần cuối cùng
- **$arrayElemAt: [{ $split: [...] }, -1] : Lây phần cuối cùng của URL đã phân tách**
- **$split: ["last-part", "?"] : Phần lấy ra mang đi phân tách bởi dấu “?”**

...

Lọc product_id và url ra 1 collection riêng để thực hiện crawl dựa trên url sau

```
db.summary.aggregate([
  {
    $match: {
      product_id: { $exists: true }
    }
  },
  {
    $group: {
      _id: "$product_id",
      urls: { $push: "$current_url" }
    }
  },
  {
    $project: {
      _id: 0,
      product_id: "$_id",
      urls: { $slice: ["$urls", 1] }
    }
  },
  {
    $merge: {
      into: "product_urls_limited"
    }
  }
])
```

Dựa trên các url tương ứng với product_id thì sẽ cào tên sản phẩm và xuất file bao gồm product_id và product_name tương ứng

```
from bs4 import BeautifulSoup
import csv
import logging
import threading
import asyncio
from concurrent.futures import ThreadPoolExecutor, as_completed
from playwright.async_api import async_playwright, TimeoutError as PlaywrightTimeou
from pymongo import MongoClient
```

```
from pymongo.errors import ServerSelectionTimeoutError, ConnectionFailure
from config import mongo_port, db_name, product_id_and_url_collection
import psutil
import time
import random

# Cấu hình logging
logging.basicConfig(filename='crawl_log.txt', level=logging.INFO, format='%(asctime)

FAILED_URLS_FILE = "failed_urls.txt"
MAX_RETRIES = 2
TIMEOUT = 60000  # 60 giây timeout
DELAY_BETWEEN_REQUESTS = 1.5  # Độ trễ giữa các yêu cầu
MONGO_CONNECTION_RETRIES = 3
MONGO_RETRY_DELAY = 2  # Giây chờ giữa các lần thử kết nối MongoDB
MAX_THREADS = 7  # Số luồng tối đa
MEMORY_THRESHOLD = 90  # Ngưỡng bộ nhớ (%)

# Danh sách user-agent để xoay vòng
USER_AGENTS = [
    "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Geck
    "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like C
    "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:125.0) Gecko/20100101 Firefox/125.0
]

# Khóa để đồng bộ ghi file
csv_lock = threading.Lock()
failed_lock = threading.Lock()

def connect_to_mongo(mongo_port, db_name, collection_name):
    for attempt in range(1, MONGO_CONNECTION_RETRIES + 1):
        try:
            client = MongoClient(
                mongo_port,
                serverSelectionTimeoutMS=30000,
                connectTimeoutMS=30000,
                socketTimeoutMS=30000
            )
```

```
            client.admin.command('ping')
            db = client[db_name]
            collection = db[collection_name]
            logging.info("✅ Kết nối thành công tới MongoDB")
            return collection
        except (ServerSelectionTimeoutError, ConnectionFailure) as e:
            logging.error(f"❌ Lỗi kết nối MongoDB (lần thử {attempt}/{MONGO_CONNECT
            if attempt == MONGO_CONNECTION_RETRIES:
                raise Exception("❌ Không thể kết nối tới MongoDB sau nhiều lần thử")
            time.sleep(MONGO_RETRY_DELAY)
        except Exception as e:
            logging.error(f"❌ Lỗi MongoDB không xác định: {str(e)}")
            raise

async def get_product_name_async(url, retries=MAX_RETRIES):
    for attempt in range(1, retries + 1):
        browser = None
        page = None
        try:
            # Kiểm tra bộ nhớ
            memory_percent = psutil.virtual_memory().percent
            if memory_percent > MEMORY_THRESHOLD:
                logging.error(f"❌ Bộ nhớ quá cao ({memory_percent}%) tại URL: {url}. Dừng
                return None

            async with async_playwright() as p:
                browser = await p.chromium.launch(headless=True)
                page = await browser.new_page(user_agent=random.choice(USER_AGENTS)
                logging.info(f"🌐 Thử {attempt}/{retries} cho URL: {url} (Bộ nhớ: {memory_pe
                await page.goto(url, timeout=TIMEOUT, wait_until="domcontentloaded")
                content = await page.content()
                soup = BeautifulSoup(content, "html.parser")

                # Tìm tên sản phẩm, ưu tiên span.base
                span_title = soup.find("span", class_="base", attrs={"data-ui-id": "page-title-
                if span_title and span_title.text.strip():
                    return span_title.text.strip()
```

```
            h1 = soup.find("h1")
            if h1 and h1.text.strip():
                return h1.text.strip()

            title = soup.find("title")
            if title and title.text.strip():
                return title.text.strip()

            meta = soup.find("meta", attrs={"name": "title"})
            if meta and meta.get("content"):
                return meta["content"].strip()

            logging.info(f"⚠️ Không tìm thấy tên sản phẩm cho {url}")
            return None
        except PlaywrightTimeoutError as e:
            logging.error(f"❌ Lỗi timeout (thử {attempt}/{retries}) cho {url}: {e}")
            if attempt == retries:
                logging.error(f"❌ Đạt tối đa số lần thử cho {url}")
                return None
            await asyncio.sleep(attempt * 2)
        except Exception as e:
            logging.error(f"❌ Lỗi (thử {attempt}/{retries}) cho {url}: {e}")
            if attempt == retries:
                logging.error(f"❌ Đạt tối đa số lần thử cho {url}")
                return None
            await asyncio.sleep(attempt * 2)
        finally:
            if page:
                await page.close()
            if browser:
                await browser.close()
            await asyncio.sleep(DELAY_BETWEEN_REQUESTS)

def get_product_name(url):
    # Chạy tác vụ async trong luồng
    loop = asyncio.new_event_loop()
    asyncio.set_event_loop(loop)
    try:
```

```
        result = loop.run_until_complete(get_product_name_async(url))
        return result
    finally:
        loop.close()

def crawl_urls(docs):
    results = []
    with ThreadPoolExecutor(max_workers=MAX_THREADS) as executor:
        future_to_url = {}
        for doc in docs:
            product_id = doc.get("product_id")
            urls = doc.get("urls", [])
            for url in urls:
                future = executor.submit(get_product_name, url)
                future_to_url[future] = (product_id, url)

        for future in as_completed(future_to_url):
            product_id, url = future_to_url[future]
            try:
                result = future.result()
                results.append((product_id, url, result))
            except Exception as e:
                logging.error(f"❌ Lỗi luồng cho {url}: {e}")
                results.append((product_id, url, None))
    return results

def main():
    try:
        collection = connect_to_mongo(mongo_port, db_name, product_id_and_url_collec
        logging.info(f"📦 Tổng số tài liệu trong collection: {collection.count_documents({}

        # Lấy tối đa 10 tài liệu, chỉ lấy trường cần thiết
        docs = list(collection.find({}, {"product_id": 1, "urls": 1}).limit(1000))

        # Crawl các URL
        results = crawl_urls(docs)

        # Ghi kết quả
```

```
        with open("product_names.csv", mode="w", newline="", encoding="utf-8") as cs
            writer = csv.writer(csvfile)
            writer.writerow(["product_id", "product_name"])

            for product_id, url, result in results:
                if isinstance(result, str) and result:
                    with csv_lock:
                        writer.writerow([product_id, result])
                    logging.info(f"✅ Tìm thấy: {result} cho product_id: {product_id}")
                else:
                    with failed_lock:
                        failed_file.write(f"{product_id},{url}\n")
                    logging.info(f"❌ Không tìm thấy tên sản phẩm cho URL: {url}")

        logging.info("🎉 HOÀN TẤT ghi tên sản phẩm vào file.")
    except Exception as e:
        logging.error(f"❌ Lỗi nghiêm trọng trong main: {str(e)}")
        raise

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

Store the data in CSV file(s) for later transformation

Bước trên đã xuất file .csv lưu tên và product_id

7) Documentation & Testing

Documents setup process

Verify data quality

- **Kiểm tra tính đầy đủ**

Tính số lượng các Missing value trên các field như $time_stamp, $ip, $user_id_db bị missing

```
db.summary_with_location.aggregate([
 { $match: {
    $or: [
     { "time_stamp": null },
     { "ip": null },
     { "user_id_db": null },
     { "time_stamp": "" },
     { "ip": "" },
     { "user_id_db": "" }
    ]
 }},
 { $count: "missing_fields_count" }
])

[ { missing_fields_count: 39440191 } ]
```

Tính số lượng ip bị thiếu or NULL trên từng fields

```
countly> db.summary_with_location.aggregate([
...   {
...     $group: {
...       _id: null,
...       missing_time_stamp: { $sum: { $cond: [{ $or: [ { $eq: ["$time_stamp", null] },
...       missing_ip: { $sum: { $cond: [{ $or: [ { $eq: ["$ip", null] }, { $eq: ["$ip", ""] } ]
...       missing_user_id_db: { $sum: { $cond: [{ $or: [ { $eq: ["$user_id_db", null] }, {
...     }
...   }
... ])

[
  {
    _id: null,
    missing_time_stamp: 0,
    missing_ip: 0,
    missing_user_id_db: 39440191
```

```
  }
]
```

Kiểm tra device_id có bị thiếu không

```
countly> db.summary_with_location.find({ "device_id": { $exists: false } }).count()
0
```

- **Kiểm tra tính chính xác**

  Kiểm tra định dạng email

```
countly> db.summary_with_location.countDocuments({
...   "email_address": {
...     $not: { $regex: /^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA/^+@+\.$/ }
...   }
... })
39443921
```

  Đảm bảo địa chỉ IP có định dạng đúng (4 số từ 0-255, phân tách bằng dấu chấm).

```
countly> db.summary_with_location.aggregate([
...   { $match: {
...       "ip": {
...         $not: /^(?:(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.){3}(?:25[0-5]|/^(?:25[0
...       }
...   }},
...   { $count: "invalid_ip_count" }
... ])
[ { invalid_ip_count: 4015 } ]
```

- **Kiểm tra tính hợp lệ**

  Resolution: Đếm số lượng Resolution không hợp lệ

```
countly> db.summary_with_location.countDocuments({
...   "resolution": {
```

```
...     $not: { $regex: /^\d{2,5}x/^\dx\d$/ } }
... })

479
```

- Kiểm tra tính duy nhất

  danh sách các `device_id` có nhiều hơn 1 lần xuất hiện và số lần xuất hiện của chúng

```
countly> db.summary_with_location.aggregate([
...   { $group: { _id: "$device_id", count: { $sum: 1 } } },
...   { $match: { count: { $gt: 1 } } },
...   { $sort: { count: -1 } }  // Sắp xếp theo số lần trùng giảm dần
... ])
[
  { _id: '741b47de-f59b-460c-aa0d-6a4a54bc42ef', count: 20902 },
  { _id: '89543d1a-4475-45d2-9f83-2d10fa51c24a', count: 14021 },
  { _id: '8134ace8-90e8-4419-887d-328bdceadabc', count: 10327 },
  { _id: '9bb3338b-578d-4ed6-b61c-33c5840d4d75', count: 10044 },
  { _id: 'aa253482-3f6f-4979-9d34-a8c415a9ce29', count: 9057 },
  { _id: 'd2e380d9-1a61-4618-a42b-0d302d42d910', count: 8820 },
  { _id: 'ccae8527-1bce-45cc-935f-a83e1e400903', count: 7995 },
  { _id: 'f12d7a98-f99c-422d-918c-4205aa314651', count: 7457 },
  { _id: '13281e27-66bd-4227-9fbb-312a9792f10d', count: 7301 },
  { _id: '97fc4315-ab81-42fe-9977-7a856a8ffaba', count: 6860 },
  { _id: '416c6834-f684-4e17-9db9-99c6cefff93b', count: 6311 },
  { _id: '558b4619-cb70-420c-8ea9-7bc19afae04d', count: 6023 },
  { _id: 'd047a20b-b513-4111-a751-1549b131e32d', count: 5805 },
  { _id: 'b10508c5-93d3-49f3-809a-380d9dc5e06e', count: 5793 },
  { _id: '78bad328-88d5-4223-b583-7e5c12816bb5', count: 5688 },
  { _id: 'b80338d1-d613-4f0e-8acb-9f68e8c0bb30', count: 5620 },
  { _id: 'e3fabc48-7b4a-4516-a078-7e987f970e1e', count: 5604 },
  { _id: '420162b0-a981-4ed1-84fc-72ef2228a8e5', count: 5494 },
  { _id: 'bd3bef0d-5758-485c-8616-cc2aaa0cef6a', count: 5435 },
  { _id: '46a34d18-1d2b-4344-a6b0-7290bcb12512', count: 5396 }
]
```

Do the data profiling on dataset

🏋️ **Data Profiling** may contain these steps:

- Gather information about the data sources, such as table names, column names, data types, constraints, and relationships between tables.
- **C**ount Null Values, Distinct Values, Data Types Consistency check.