CB??????

https://cbreport.zfun.com.tw/

CB??????

? ??

???????? **Flask** ??????????????????GA4 ???????????
???????????????????????????? Excel ???

? ????

1. **???????**
   - ???? `/root/cb_order/orders` ?? JSON ?????
   - ????????????
2. **GA4 ????????**
   - ???? GA4 ??????? (`ga4_YYYYMMDD.json`)?
   - ????????????
3. **?????**
   - ?? `new_old_YYYYMMDD.json` ????????????
   - ????????????
4. **??????**
   - ?? `ga4_s_YYYYMMDD.json`??????? (sessionSourceMedium)?
5. **?????**
   - ??????
   - ????
   - ??????
   - ???????????????
6. **Excel ??**
   - ?????? / ?? / ?????? (`export.xlsx`)?
   - ????????????????????? (`export_new_old.xlsx`)?

? ??????

- ?????`/root/cb_order/orders/`
  - ???`20250101.json`?`ga4_20250101.json`?`new_old_20250101.json`
- ?????`/root/cb_p/transformed_products.json`
- ?????
  - `/tmp/order_summary_export.xlsx`
  - `/tmp/new_old_export.xlsx`

? ??????

`export.xlsx`

- **??**
- **??**
- **??**

`export_new_old.xlsx`

- ???
- ???
- ????? / ?? / ??
- ????? / ??
- ???
- ???
- ????
- ??

?? ????

1. ????

```
python app.py
```

1. ??????? `http://localhost:5007`
2. ???????
   - ???????????
   - ?????????
3. ????
   - [????] ? `export.xlsx`
   - [???????] ? `export_new_old.xlsx`

? ????

- ????????
- ?????????
- ??????
- ???? / ???? / ??????
- ?????????
- GA4 ??????

?? ?????

```
import os
import json
from flask import Flask, jsonify, render_template, request, send_file
from glob import glob
from datetime import datetime, timedelta
from collections import defaultdict
import pandas as pd

app = Flask(__name__)
ORDER_DIR = "/root/cb_order/orders"
EXPORT_FILE = "/tmp/order_summary_export.xlsx"
NEW_OLD_EXPORT_FILE = "/tmp/new_old_export.xlsx"
PRODUCTS_PATH = "/root/cb_p/transformed_products.json"

def load_orders():
    orders = []
    for file_path in glob(os.path.join(ORDER_DIR, "*.json")):
        if "ga4_" in file_path or "new_old_" in file_path:
            continue
        try:
            with open(file_path, "r", encoding="utf-8") as f:
                data = json.load(f)
                if isinstance(data, list):
                    orders.extend(data)
                elif isinstance(data, dict):
                    orders.append(data)
        except Exception as e:
            print(f"□ □□□□ {file_path}: {e}")
    return orders

def filter_orders_by_date(orders, start_date, end_date):
    def parse_time(ts):
        try:
            return datetime.strptime(ts, "%Y-%m-%d %H:%M:%S")
        except (ValueError, TypeError):
            return None
```

```
    start_dt = datetime.strptime(start_date, "%Y-%m-%d") if start_date else datetime.min
    end_dt = datetime.strptime(end_date, "%Y-%m-%d") if end_date else datetime.max
    end_dt = end_dt.replace(hour=23, minute=59, second=59)

    filtered = []
    for order in orders:
        try:
            created = parse_time(order.get("created_at"))
            if created and start_dt <= created <= end_dt:
                filtered.append(order)
        except Exception as e:
            print(f"⚠️ □□□□□□□□□: {e}")
            continue
    return filtered

def load_ga4_traffic(start_date, end_date):
    if not start_date or not end_date:
        return 0, []
    try:
        start_dt = datetime.strptime(start_date, "%Y-%m-%d")
        end_dt = datetime.strptime(end_date, "%Y-%m-%d")
    except Exception:
        return 0, []
    total_traffic = 0
    daily_traffic = []
    cur_date = start_dt
    while cur_date <= end_dt:
        date_str = cur_date.strftime("%Y%m%d")
        json_path = os.path.join(ORDER_DIR, f"ga4_{date_str}.json")
        count = 0
        try:
            with open(json_path, "r", encoding="utf-8") as f:
                arr = json.load(f)
                if isinstance(arr, list) and arr:
                    count = int(arr[0].get("activeUsers", 0))
        except Exception:
            pass
        daily_traffic.append({"date": date_str, "activeUsers": count})
        total_traffic += count
        cur_date += timedelta(days=1)
```

```
    return total_traffic, daily_traffic

def load_new_old_data(start_date, end_date):
    new_old_map = {}
    if not start_date or not end_date:
        return new_old_map
    try:
        start_dt = datetime.strptime(start_date, "%Y-%m-%d")
        end_dt = datetime.strptime(end_date, "%Y-%m-%d")
    except Exception as e:
        print(f"⚠□ □□□□□□: {e}")
        return new_old_map
    cur_date = start_dt
    while cur_date <= end_dt:
        date_str = cur_date.strftime("%Y%m%d")
        json_path = os.path.join(ORDER_DIR, f"new_old_{date_str}.json")
        try:
            with open(json_path, "r", encoding="utf-8") as f:
                data = json.load(f)
                if isinstance(data, list):
                    for entry in data:
                        order_name = entry.get("order_name")
                        newold = entry.get("newold")
                        if order_name and newold in ["new", "old"]:
                            new_old_map[order_name] = newold
        except Exception as e:
            print(f"⚠□ □□□□□□□□□ {json_path}: {e}")
        cur_date += timedelta(days=1)
    return new_old_map

def load_ga4_source_data(start_date, end_date):
    source_map = {}
    if not start_date or not end_date:
        return source_map
    try:
        start_dt = datetime.strptime(start_date, "%Y-%m-%d")
        end_dt = datetime.strptime(end_date, "%Y-%m-%d")
    except Exception as e:
        print(f"⚠□ □□□□□□: {e}")
        return source_map
```

```
    cur_date = start_dt
    while cur_date <= end_dt:
        date_str = cur_date.strftime("%Y%m%d")
        json_path = os.path.join(ORDER_DIR, f"ga4_s_{date_str}.json")
        try:
            with open(json_path, "r", encoding="utf-8") as f:
                data = json.load(f)
                if isinstance(data, list):
                    for entry in data:
                        transaction_id = entry.get("transaction_id")
                        session_source_medium = entry.get("sessionSourceMedium")
                        if transaction_id and session_source_medium:
                            source_map[transaction_id] = session_source_medium
        except Exception as e:
            print(f"⚠□ □□□□□□□□□□□ {json_path}: {e}")
        cur_date += timedelta(days=1)
    return source_map

def analyze_summary(orders, start_date="", end_date=""):
    total_orders = len(orders)
    total_amount = sum(order.get("prices", {}).get("total_price", 0) for order in orders)

    normal_products = {}
    gifts = {}
    payment_methods = defaultdict(int)
    device_types = defaultdict(int)
    source_types = defaultdict(int)
    cities = defaultdict(int)

    new_old_map = load_new_old_data(start_date, end_date)
    source_map = load_ga4_source_data(start_date, end_date)

    new_customer_count = 0
    old_customer_count = 0
    new_customer_amount = 0.0
    old_customer_amount = 0.0

    new_old_data = []  # □□□□□□□□□□

    for order in orders:
```

```
method = order.get("payment_method", "□□")
device = order.get("from_device", "□□")
order_name = order.get("order_name", "□□")
session_source_medium = source_map.get(order_name, "□□")
payment_methods[method] += 1
device_types[device] += 1
source_types[session_source_medium] += 1

try:
    customer = order.get("customer")
    if customer and isinstance(customer, dict):
        addr = customer.get("address")
        if addr and isinstance(addr, dict):
            detail = addr.get("detail_address")
            if detail and isinstance(detail, dict):
                city = detail.get("city")
                if city:
                    cities[city] += 1
except Exception as e:
    print(f"⚠□ □□□□□□□: {e}")

try:
    amount = order.get("prices", {}).get("total_price", 0)
    customer = order.get("customer", {})
    name = customer.get("name", "□□") if isinstance(customer, dict) else "□□"
    email = customer.get("email", "□□") if isinstance(customer, dict) else "□□"
    mobile = customer.get("mobile", "□□") if isinstance(customer, dict) else "□□"
    created_at = order.get("created_at", "□□")

    newold = new_old_map.get(order_name, "new")
    if newold == "new":
        new_customer_count += 1
        new_customer_amount += amount
    elif newold == "old":
        old_customer_count += 1
        old_customer_amount += amount

    new_old_data.append({
        "order_name": order_name,
        "created_at": created_at,
```

```
            "customer_name": name,
            "customer_email": email,
            "customer_mobile": mobile,
            "total_price": amount,
            "newold": newold,
            "session_source_medium": session_source_medium
        })
    except Exception as e:
        print(f"⚠□ □□□□□□□□□□□□□□: {e}")

    for item in order.get("line_items", []):
        name = item.get("title", "")
        spec = item.get("variant_title", "")
        full_name = f"{name} {spec}".strip()
        qty = item.get("quantity", 0)
        item_type = item.get("item_type", "normal")

        if item_type in ["gift", "first-order-gift"]:
            gifts[full_name] = gifts.get(full_name, 0) + qty
        else:
            normal_products[full_name] = normal_products.get(full_name, 0) + qty

top_products = sorted(normal_products.items(), key=lambda x: x[1], reverse=True)
top_gifts = sorted(gifts.items(), key=lambda x: x[1], reverse=True)

def format_count(d): return ", ".join(f"{k}: {v}" for k, v in d.items())

total_traffic, daily_traffic = load_ga4_traffic(start_date, end_date)
conversion_rate = (total_orders / total_traffic * 100) if total_traffic > 0 else 0

return {
    "total_orders": total_orders,
    "total_amount": total_amount,
    "top_products": top_products,
    "top_gifts": top_gifts,
    "payment_summary": format_count(payment_methods),
    "device_summary": format_count(device_types),
    "source_summary": format_count(source_types),
    "city_summary": format_count(cities),
    "new_customer_count": new_customer_count,
```

```
        "old_customer_count": old_customer_count,
        "new_customer_amount": new_customer_amount,
        "old_customer_amount": old_customer_amount,
        "total_traffic": total_traffic,
        "conversion_rate": conversion_rate,
        "daily_traffic": daily_traffic,
        "new_old_data": new_old_data,
        "raw": {
            "cities": cities,
            "products": normal_products,
            "gifts": gifts,
        }
    }

@app.route("/")
def index():
    start_date = request.args.get("start_date", "")
    end_date = request.args.get("end_date", "")
    summary = None

    if start_date and end_date:
        orders = load_orders()
        filtered = filter_orders_by_date(orders, start_date, end_date)
        summary = analyze_summary(filtered, start_date, end_date)

        df_city = pd.DataFrame(list(summary["raw"]["cities"].items()), columns=["□□", "□□"])
        df_prod = pd.DataFrame(list(summary["raw"]["products"].items()), columns=["□□", "□□"])
        df_gift = pd.DataFrame(list(summary["raw"]["gifts"].items()), columns=["□□", "□□"])
        with pd.ExcelWriter(EXPORT_FILE) as writer:
            df_city.to_excel(writer, sheet_name="□□", index=False)
            df_prod.to_excel(writer, sheet_name="□□", index=False)
            df_gift.to_excel(writer, sheet_name="□□", index=False)

    return render_template("index.html", summary=summary, start_date=start_date,
end_date=end_date)

@app.route("/export.xlsx")
def download_excel():
    return send_file(EXPORT_FILE, as_attachment=True)
```

```
@app.route("/export_new_old.xlsx")
def download_new_old_excel():
    start_date = request.args.get("start_date", "")
    end_date = request.args.get("end_date", "")
    if not start_date or not end_date:
        return jsonify({"error": "請提供起始日期和結束日期"}), 400

    orders = load_orders()
    filtered = filter_orders_by_date(orders, start_date, end_date)
    if not filtered:
        return jsonify({"error": "指定日期範圍內無訂單資料"}), 404

    summary = analyze_summary(filtered, start_date, end_date)
    if not summary["new_old_data"]:
        return jsonify({"error": "指定日期範圍內無新舊客資料"}), 404

    # 載入 product_id 對應溫層資料
    product_temp_map = {}
    try:
        with open(PRODUCTS_PATH, encoding="utf-8") as f:
            plist = json.load(f)
            if isinstance(plist, list):
                for prod in plist:
                    pid = prod.get("product_id")
                    temp = prod.get("temp", [])
                    if pid is not None:
                        product_temp_map[int(pid)] = temp
            elif isinstance(plist, dict):
                pid = plist.get("product_id")
                temp = plist.get("temp", [])
                if pid is not None:
                    product_temp_map[int(pid)] = temp
    except Exception as e:
        print(f"□ □□ transformed_products.json 失敗: {e}")

    new_old_list = []
    for order in filtered:
        order_name = order.get("order_name", "")
        created_at = order.get("created_at", "")
        customer = order.get("customer", {}) or {}
```

```
buyer = order.get("buyer", {}) or {}
receiver = order.get("receiver", {}) or {}
prices = order.get("prices", {}) or {}

# newold/session_source_medium
newold = "未知"
session_source_medium = "未知"
for entry in summary["new_old_data"]:
    if entry["order_name"] == order_name:
        newold = entry["newold"]
        session_source_medium = entry["session_source_medium"]
        break

# 判斷訂單溫層
all_temps = []
for item in order.get("line_items", []):
    pid = item.get("product_id")
    if pid is not None:
        temp = product_temp_map.get(int(pid))
        if temp and isinstance(temp, list):
            all_temps.extend(temp)
# 全為常溫則常溫，否則冷凍
if all_temps and all(t == "常溫" for t in all_temps):
    order_temp = "常溫"
else:
    order_temp = "冷凍"

new_old_list.append({
    "訂單號": order_name,
    "訂購日": created_at,
    "寄件人姓名": customer.get("name", ""),
    "寄件人信箱": buyer.get("email", ""),
    "寄件人電話": buyer.get("mobile", ""),
    "收件人姓名": receiver.get("name", ""),
    "收件人電話": receiver.get("phone", ""),
    "總金額": prices.get("total_price", 0),
    "新舊客": newold,
    "訂單來源": session_source_medium,
    "溫層": order_temp,
})
```

```
    df_new_old = pd.DataFrame(new_old_list, columns=[
        "訂單號", "訂購日", "寄件人姓名", "寄件人信箱", "寄件人電話",
        "收件人姓名", "收件人電話", "總金額", "新舊客", "訂單來源", "溫層"
    ])

    with pd.ExcelWriter(NEW_OLD_EXPORT_FILE) as writer:
        df_new_old.to_excel(writer, sheet_name="新舊客資料", index=False)

    return send_file(NEW_OLD_EXPORT_FILE, as_attachment=True)

if __name__ == "__main__":
    app.run(debug=True, port=5007)
```

CB??????????

? ??

??????? **Cyberbiz API** ???????????????? **Gmail API** ???????????????? LINE ??????????????

? ????

1. **??????**
   - ?? Cyberbiz API (`/v1/orders`) ??????????open??
   - ??????????????
2. **????**
   - ?? JSON ?????? `/root/cb_order/orders/cb_order_YYYYMMDD.json`?
3. **????**
   - ???
     - ????
     - ?????
     - ???????
     - ??????
     - ????
4. **Email ????**
   - ?? Gmail API?OAuth ????????
   - ?????

```
□□□□□□□ - YYYYMMDD
```

   - ?????
     - ????
     - ??????????
     - ?????
     - ??????
     - ????????
2. **LINE ????????**
   - ?? LINE Bot API ??????????????
   - ?????????????

? ???????

- **??????**
  - `/root/cb_order`
- **??????**
  - `/root/cb_order/orders/cb_order_YYYYMMDD.json`

- **Gmail ??**
  - `credentials.json` ?Gmail OAuth ??
  - `token.json` ???????????
- **????**
  - `BASE_URL` ?Cyberbiz API ??
  - `TOKEN` ?Cyberbiz API Token
  - `TO_EMAILS` ??????
  - `SENDER` ???? Gmail ??

? ????

□□銷售日期: 20250331

□□訂單數量: 18【信用卡:12、LINE Pay:5、貨到付款:1】

□□總銷售金額: 24500 元

□□銷售商品明細:

- 綜合禮盒: 8 件
- 招牌魚鬆: 6 件

□□贈品明細:

- 試吃包: 4 件

?? ????

1. ?? `TOKEN` ? Gmail ?? (`credentials.json`)?
2. ?????

   ```
   python3 cb_daily_report.py
   ```

3. ??????
   - ??????
   - ?? JSON ??
   - ???????
   - ?? Email ??
   - ??????? LINE ??

? ??

- ??????????????
- ?????????????/????
- ???????????

?? ?????

```
#!/usr/bin/env python3
import os
import json
import base64
import requests
from datetime import datetime, timedelta
from collections import defaultdict
from email.message import EmailMessage
from google.auth.transport.requests import Request
from google.oauth2.credentials import Credentials
from google_auth_oauthlib.flow import InstalledAppFlow
from googleapiclient.discovery import build

# === 切換目錄，避免 cron 找不到 json 憑證 ===
os.chdir('/root/cb_order')

# === 設定 ===
BASE_URL = "https://app-store-api.cyberbiz.io"
TOKEN = ""
TO_EMAILS = [
    "wayne@lianhung.com.tw"
]
SENDER = "zfuntw@gmail.com"
SCOPES = ['https://www.googleapis.com/auth/gmail.send']

# === 抓昨天的資料 ===
yesterday = datetime.now() - timedelta(days=1)
start_time = yesterday.replace(hour=0, minute=0, second=0, microsecond=0)
end_time = yesterday.replace(hour=23, minute=59, second=59, microsecond=0)
start_time_str = start_time.strftime("%Y-%m-%d %H:%M:%S")
end_time_str = end_time.strftime("%Y-%m-%d %H:%M:%S")
report_date_str = yesterday.strftime("%Y%m%d")

# === 建立 orders/ 目錄與檔名 ===
os.makedirs("orders", exist_ok=True)
json_path = f"orders/cb_order_{report_date_str}.json"
```

```
# === 取得訂單清單 ===
params = {
    "start_time": start_time_str,
    "end_time": end_time_str,
    "statuses": "open"
}
HEADERS = {
    "Authorization": f"Bearer {TOKEN}",
    "Accept": "application/json"
}
response = requests.get(f"{BASE_URL}/v1/orders", headers=HEADERS, params=params)
if response.status_code != 200:
    print("□ □□□□□□□□□")
    print(response.text)
    exit()
order_list = response.json()

# === 取得每筆訂單詳情 ===
all_order_details = []
for order in order_list:
    order_id = order.get("id")
    if not order_id:
        continue
    detail_res = requests.get(f"{BASE_URL}/v1/orders/{order_id}", headers=HEADERS)
    if detail_res.status_code == 200:
        all_order_details.append(detail_res.json())
        print(f"□ □□□□□ {order_id}")
    else:
        print(f"⚠◌ 無法讀取訂單 {order_id}")

# === 儲存 JSON 檔案 ===
with open(json_path, "w", encoding="utf-8") as f:
    json.dump(all_order_details, f, ensure_ascii=False, indent=2)

# === 統計分析（付款方式 + 銷售商品 + 贈品）===
total_orders = len(all_order_details)
total_amount = 0
product_summary = defaultdict(int)
gift_summary = defaultdict(int)
payment_methods = defaultdict(int)
```

```
for order in all_order_details:
    payment = order.get("payment_method", "未知方式")
    payment_methods[payment] += 1

    for item in order.get("line_items", []):
        title = item.get("title", "未知商品")
        qty = item.get("quantity", 0)
        price = item.get("total_price_after_discounts", item.get("price", 0))
        if price == 0:
            gift_summary[title] += qty
        else:
            product_summary[title] += qty
            total_amount += price

# === 組合付款方式統計字串 ===
payment_str = "、".join([f"{k}:{v}" for k, v in sorted(payment_methods.items(), key=lambda x: -
x[1])])

# === 組合信件內容 ===
report_lines = [
    f"□□銷售日期: {report_date_str}",
    f"□□訂單數量: {total_orders}（{payment_str}）",
    f"□□總銷售金額: {total_amount:.0f} 元",
    "",
    "□□銷售商品明細: "
]
for title, qty in sorted(product_summary.items(), key=lambda x: -x[1]):
    report_lines.append(f"- {title}: {qty} 件")

if gift_summary:
    report_lines.append("")
    report_lines.append("□□贈品明細: ")
    for title, qty in sorted(gift_summary.items(), key=lambda x: -x[1]):
        report_lines.append(f"- {title}: {qty} 件")

email_body = "\n".join(report_lines)

# === Gmail API 驗證 ===
creds = None
```

```
if os.path.exists('token.json'):
    creds = Credentials.from_authorized_user_file('token.json', SCOPES)
if not creds or not creds.valid:
    if creds and creds.expired and creds.refresh_token:
        creds.refresh(Request())
    else:
        flow = InstalledAppFlow.from_client_secrets_file('credentials.json', SCOPES)
        creds = flow.run_local_server(port=0)
    with open('token.json', 'w') as token:
        token.write(creds.to_json())

service = build('gmail', 'v1', credentials=creds)

# === 寄出 Email ===
message = EmailMessage()
message.set_content(email_body)
message['To'] = ", ".join(TO_EMAILS)
message['From'] = SENDER
message['Subject'] = f"□□銷售報表 - {report_date_str}"

encoded_message = base64.urlsafe_b64encode(message.as_bytes()).decode()
send_message = service.users().messages().send(userId="me", body={"raw":
encoded_message}).execute()

print(f"\n□ □□□□□□ID: {send_message['id']}")

# === 發送 LINE 群組訊息 ===
#from linebot import LineBotApi
#from linebot.models import TextSendMessage
#from dotenv import load_dotenv

# 載入環境變數（如果程式開頭已有則可略過）
#load_dotenv()

#print("DEBUG - LINE_CHANNEL_ACCESS_TOKEN:", os.getenv("LINE_CHANNEL_ACCESS_TOKEN"))
#print("DEBUG - LINE_GROUP_IDS:", os.getenv("LINE_GROUP_IDS"))

#LINE_TOKEN = os.getenv("LINE_CHANNEL_ACCESS_TOKEN")
#LINE_GROUP_IDS = os.getenv("LINE_GROUP_IDS", "").split(",")
#LINE_GROUP_IDS = [gid.strip() for gid in LINE_GROUP_IDS if gid.strip()]
```

```
#if LINE_TOKEN and LINE_GROUP_IDS:
#    try:
#        line_bot_api = LineBotApi(LINE_TOKEN)

        # LINE 最多允許 500 字，過長會送不出去
#        short_report = "\n".join(report_lines)
#        max_len = 480
#        if len(short_report) > max_len:
#            short_report = short_report[:max_len] + "\n...（訊息過長略）"
#
#        for gid in LINE_GROUP_IDS:
#            try:
#                line_bot_api.push_message(
#                    gid,
#                    TextSendMessage(text=short_report)
#                )
#                print(f"□ □□□ LINE 銷售報表給群組: {gid}")
#            except Exception as e:
#                print(f"□ □□ LINE 給 {gid} 失敗: {e}")
#
#    except Exception as e:
#        print(f"□ □□ LINE 失敗（初始化錯誤）: {e}")
#else:
#    print("⚠◌ 未設定 LINE_CHANNEL_ACCESS_TOKEN 或 LINE_GROUP_IDS")
```