LAB 2: DATA INTEGRATION - THE ETL PIPELINE

A. THÔNG TIN CHUNG (META INFO)

- **Môn học:** CMU-CS 445 System Integration Practices.
- **Ánh xạ kiến thức:** Lecture 3 (Data Integration) - Data Warehousing & ETL.
- **Thời lượng ước tính:** 75 phút.
- **Cấp độ:** Trung bình (Intermediate).

B. MỤC TIÊU & BỐI CẢNH (OBJECTIVES & SCENARIO)

1. Kịch bản (Scenario)

Phòng Marketing của NOAH Retail đang cần danh sách "Top Khách hàng chi tiêu nhiều nhất" để gửi quà tri ân.

Tuy nhiên, dữ liệu đang nằm rải rác ở 2 nơi:

1. **Thông tin cá nhân (Tên, Email):** Nằm trong một file Text cũ (`customers.csv`) xuất từ hệ thống CRM.
2. **Lịch sử giao dịch:** Nằm trong Database bán hàng (`orders.db` - SQLite).

Nếu làm thủ công, bạn phải mở Excel lên và VLOOKUP rất vất vả. Nhiệm vụ của bạn là viết một chương trình Python tự động hóa quy trình này: Đọc dữ liệu từ 2 nguồn khác nhau -> Gộp lại -> Tính toán tổng chi tiêu.

2. Mục tiêu kỹ thuật

- Hiểu quy trình **ETL**: **E**xtract (Trích xuất) -> **T**ransform (Xử lý) -> **L**oad (Tải ra).
- Sử dụng thư viện **Pandas** để xử lý dữ liệu dạng bảng.
- Kỹ thuật **Data Stitching**: Ghép nối dữ liệu từ các nguồn không đồng nhất (Heterogeneous Sources).

3. Sơ đồ kiến trúc (Architecture)

C. CHUẨN BỊ (PREREQUISITES)

1. **Môi trường:** Đã cài đặt Docker và VS Code.
2. **Tài nguyên:** Tạo thư mục `Lab2_ETL` trên máy tính.

D. CÁC BƯỚC THỰC HIỆN (STEP-BY-STEP)

TASK 1: CHUẨN BỊ DỮ LIỆU GIẢ LẬP (PREPARE DATA)

Mục đích: Tạo ra 2 nguồn dữ liệu rời rạc để thực hành ghép nối.

Bước 1.1: Trong thư mục `Lab2_ETL`, tạo file `customers.csv` (Nguồn 1):
Code

```
id,name,email
1,Nguyen Van A,anv@gmail.com
2,Tran Thi B,btt@yahoo.com
3,Le Van C,cle@hotmail.com
4,Pham Van D,dpham@gmail.com
```

Bước 1.2: Tạo file `init_db.py` để sinh Database SQLite (Nguồn 2):
Python

```
import sqlite3

# Tạo file database giả lập
conn = sqlite3.connect('orders.db')
c = conn.cursor()

# Tạo bảng Orders
c.execute('''CREATE TABLE orders
             (order_id INTEGER PRIMARY KEY, customer_id INTEGER, amount
REAL)''')

# Insert dữ liệu mẫu (Khách hàng 1 mua 2 lần, Khách 2 mua 1 lần, Khách 3
không mua)
data = [
    (101, 1, 500.0),
    (102, 1, 300.0),
    (103, 2, 1200.0),
    (104, 4, 150.0)
]
c.executemany('INSERT INTO orders VALUES (?,?,?)', data)

conn.commit()
conn.close()
print("Database orders.db created successfully!")
```

Bước 1.3: Chạy file này một lần để tạo file .db:

(Nếu máy chưa cài Python, bạn có thể bỏ qua bước chạy này, chúng ta sẽ để Docker chạy nó sau).

TASK 2: VIẾT ETL SCRIPT (THE INTEGRATOR)

Mục đích: Viết logic "Trái tim" của hệ thống tích hợp.

Tạo file `etl_script.py` với nội dung sau:
Python

```python
import pandas as pd
import sqlite3
import os

def run_etl():
    print(" STARTED ETL PROCESS...")

    # --- PHASE 1: EXTRACT (Trích xuất) ---
    print("[1] Extracting data from heterogeneous sources...")

    # Nguồn 1: Đọc file CSV
    try:
        df_customers = pd.read_csv('customers.csv')
        print(f"   -> Loaded {len(df_customers)} customers from CSV.")
    except Exception as e:
        print(f"Error reading CSV: {e}")
        return

    # Nguồn 2: Đọc SQL Database
    try:
        conn = sqlite3.connect('orders.db')
        df_orders = pd.read_sql_query("SELECT * FROM orders", conn)
        print(f"   -> Loaded {len(df_orders)} orders from SQLite.")
    except Exception as e:
        print(f"Error reading DB: {e}")
        return
    finally:
        if conn: conn.close()

    # --- PHASE 2: TRANSFORM (Xử lý & Gộp) ---
    print("[2] Transforming data...")

    # Bước 2.1: Join 2 bảng lại với nhau dựa trên ID
    # (Tương tự VLOOKUP trong Excel hoặc JOIN trong SQL)
    merged_df = pd.merge(df_orders, df_customers, left_on='customer_id',
right_on='id', how='left')

    # Bước 2.2: Tính tổng tiền chi tiêu theo từng khách
    report_df = merged_df.groupby(['name',
'email'])['amount'].sum().reset_index()

    # Bước 2.3: Đổi tên cột cho đẹp
    report_df.columns = ['Customer Name', 'Email', 'Total Spent']

    # --- PHASE 3: LOAD (Tải ra báo cáo) ---
    print("[3] Loading Data to Report...")
    print("\n--- FINAL REPORT ---")
    print(report_df)

    # Xuất ra file kết quả
    report_df.to_csv('final_report.csv', index=False)
```

```
    print("\n Report saved to 'final_report.csv'")

if __name__ == "__main__":
    # Kiểm tra xem DB có tồn tại chưa, nếu chưa thì tạo (cho trường hợp
chạy Docker lần đầu)
    if not os.path.exists('orders.db'):
        import init_db # Tự động chạy script tạo DB nếu chưa có
    run_etl()
```

Giải thích cơ chế:

- `pd.read_csv` và `pd.read_sql`: Là các "Adapter" giúp Python hiểu được dữ liệu từ 2 định dạng khác nhau.
- `pd.merge`: Là kỹ thuật "Data Stitching" (Khâu dữ liệu). Nó tìm các bản ghi có chung ID và ghép chúng lại thành một dòng duy nhất.

TASK 3: ĐÓNG GÓI MÔI TRƯỜNG (DOCKERIZE)

Mục đích: Đảm bảo script chạy được trên mọi máy mà không cần cài Pandas thủ công.

Tạo file `Dockerfile`:
Dockerfile

```
# Sử dụng Python Slim
FROM python:3.9-slim

WORKDIR /app

# Cài đặt thư viện Pandas (Xử lý dữ liệu)
# Lưu ý: Cài pandas sẽ hơi lâu (1-2 phút)
RUN pip install pandas sqlalchemy

# Copy toàn bộ code vào
COPY . .

# Chạy script tạo DB trước, sau đó chạy ETL
CMD ["python", "etl_script.py"]
```

TASK 4: CHẠY VÀ KIỂM TRA

Mục đích: Thực thi quy trình.

Bước 4.1: Build Image
Bash

```
docker build -t lab2-etl .
```

Bước 4.2: Run Container
Bash

```
docker run --name etl_worker lab2-etl
```

Bước 4.3: Kiểm tra kết quả trên Terminal. Bạn sẽ thấy bảng báo cáo hiện ra:

```
   Customer Name            Email  Total Spent
0     Le Van C    cle@hotmail.com          NaN  <-- (Nếu dùng LEFT JOIN
mà khách ko mua)
1   Nguyen Van A    anv@gmail.com        800.0
2     Pham Van D  dpham@gmail.com        150.0
3     Tran Thi B    btt@yahoo.com       1200.0
```

E. THỬ THÁCH NÂNG CAO (CHALLENGE - 20% ĐIỂM)

Tình huống: Sếp không quan tâm đến những khách hàng "tiềm năng" (chưa mua gì) hoặc mua quá ít.

Yêu cầu:

1. Sửa file `etl_script.py`.
2. Thêm logic lọc: Chỉ hiển thị những khách hàng có `Total Spent > 500`.
3. Sắp xếp danh sách theo số tiền giảm dần (Người giàu nhất đứng đầu).
4. Build lại Image và chạy lại.

Kết quả mong đợi: Báo cáo chỉ còn lại 2 người: **Tran Thi B** (1200.0) và **Nguyen Van A** (800.0).

F. HƯỚNG DẪN NỘP BÀI (SUBMISSION)

Sinh viên nộp file `Submission_RP2` gồm:

1. **Báo cáo PDF:**
   - Ảnh chụp Terminal kết quả chạy Task 4 (Bảng dữ liệu chưa lọc).
   - Ảnh chụp Terminal kết quả phần Challenge (Bảng dữ liệu đã lọc > 500).
   - Giải thích ý nghĩa lệnh `pd.merge` trong code.
2. **Source Code:** `etl_script.py`, `customers.csv`, `Dockerfile`.