Assignment 15/11 - 22/11

Ex1: Online extraction, offline extraction.

Online extraction involves directly retrieving data from source systems that are actively connected to the network. This method allows for real-time data access and processing without the need for intermediate storage.

Offline extraction refers to the process of retrieving data from sources that are not actively connected to a network. This often involves using copies of the original data stored in external locations.

Feature	Online Extraction	Offline Extraction
Connection Type	Direct connection to live source	Indirect retrieval from external copies
Staging Area Needed	No	Yes
Data Freshness	Real-time access	May contain outdated information
Typical Use Cases	Real-time monitoring, web scraping	Historical analysis, legacy system access

Use Cases Online Extraction:

- Real-Time Monitoring: Useful for applications that require immediate access to data, such as financial market analysis or social media sentiment tracking.
- Web Scraping: Commonly employed for gathering data from websites for market research, product comparisons, or competitive analysis.
- E-commerce Data Retrieval: Extracting consumer data from online shopping platforms to analyze purchasing trends.

Use Cases Offline Extraction

- Historical Data Analysis: Ideal for analyzing archived records or reports that are not frequently updated.
- Legacy System Integration: Useful for extracting data from older systems that may not be connected to the internet.
- Internal Reporting: Often used to retrieve information from internal databases or documents that are maintained offline.

Ex2: ETL, ELT. When to use? Why?

ETL (Extract, Transform, Load):

- Extract: Data is extracted from various source systems.
- Transform: The extracted data is transformed into a suitable format or structure for analysis. This can involve cleaning, filtering, aggregating, and enriching the data.
- Load: The transformed data is then loaded into a target system, typically a data warehouse or database optimized for analytical queries.

ELT (Extract, Load, Transform):

- Extract: Similar to ETL, data is extracted from source systems.
- Load: The extracted raw data is loaded directly into the target system without prior transformation.
- Transform: Once the data is in the target system, it is transformed as needed for analysis. This approach leverages the processing power of modern databases to perform transformations.

ETL is ideal when dealing with structured data from traditional databases where transformations are necessary before loading. Use ETL when there are strict requirements for data quality and integrity since transformations can be applied to clean and validate the data before it enters the warehouse. If complex transformations are needed that require significant processing before loading, ETL is more suitable.

ELT is preferable in big data scenarios where large volumes of raw data are ingested into cloud-based storage solutions like data lakes. ELT allows analysts to work with raw data and apply transformations as needed. This flexibility can lead to more dynamic reporting and analysis. With the advent of powerful cloud-based platforms (e.g., Google BigQuery, Amazon Redshift), ELT can take advantage of their scalability and processing capabilities.

Why Choose One Over the Other

- Performance Considerations: ELT can be more efficient in environments where storage costs are low and processing power is high. It reduces the time to load data since it skips the transformation step before loading.
- Data Accessibility: ELT allows users to access raw data quickly for exploratory analysis without waiting for transformations to complete. This can be beneficial in fast-paced business environments where insights need to be derived quickly.
- Simplicity of Implementation: ETL processes can be more complex due to the need for transformation logic before loading. In contrast, ELT simplifies the pipeline by loading raw data first.

Ex3: Docker exercise!

Deploy 1 hoặc nhiều docker container có thể phục vụ chạy airflow, ETL (sử dụng python, pandas…), có khả năng kết nối với datasource, cloud. Guideline:

i) Install Docker.
ii) Install python, airflow (prefect, or dagster), selenium, requests, beautifulsoup, AWS or Azure CLI (if it is too easy for you, install pyspark) on a single container, or on different containers (it depends on how you orchestrate and use these services).

Ex4: Why docker-compose?

Docker Compose is a tool for defining and running multi-container Docker applications. It allows developers to configure application services using a YAML file, simplifying the process of managing complex applications that involves multiple containers.

Benefits:

- *Simplified Configuration:* Docker Compose uses a single YAML file to define all services, networks, and volumes required for an application, making it easy to manage configurations in one place.
- *Multi-Container Management:* Define and run multiple containers as a single service, enabling easier orchestration and management.
- *Environment Consistency:* we can ensure that the application runs consistently across different environments (development, testing, production) by using the same configuration file.
- *Easier Scaling:* scale services up or down with a single command, allowing for quick adjustments based on demand without needing to modify the underlying code.
- *Integrated Networking:* Docker Compose automatically sets up a network for the containers defined in the configuration file, allowing them to communicate with each other without intervening much on local network.

Ex5: How to reduce the size of Docker images, containers?

Keyword: multistaging build.

Use Smaller Base Images

- For example: use lightweight base images like Alpine Linux, which can significantly reduce the image size compared to standard images like Ubuntu or Debian.

Minimize Layers

- Use multi-command RUN statements in your Dockerfile to reduce the number of layers.
  RUN apt-get update && apt-get install -y package1 package2 && rm -rf /var/lib/apt/lists/*
- Docker-squash to flatten image layers
  https://scalefactory.com/blog/2023/10/19/using-docker-squash-for-smaller-images/

Remove Unnecessary Files

- Clean up temporary files and caches created during the build process.
- Only include production dependencies in your final image. Use multi-stage builds to separate build-time dependencies from runtime dependencies.
- Use .dockerignore to exclude unnecessary objects added to the Docker context.

Leverage Multi-Stage Builds

- Multi-stage builds: use one Dockerfile to build your application in one stage and then copy only the necessary artifacts to a smaller base image in another stage. Thus the final image size will be minimal.

```
FROM golang:alpine AS builder
WORKDIR /app
COPY . .
RUN go build -o myapp

FROM alpine
WORKDIR /app
COPY --from=builder /app/myapp .
CMD ["./myapp"]
```