Big Data Report

Prepared by: Vu Thien Son

Date: Oct 2025

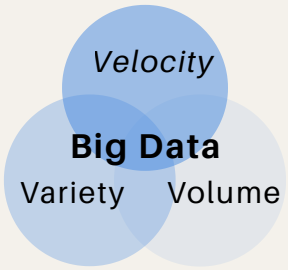
Table of Contents

01. Big Data Overview

DEFINITION

Big Data is a term used to refer to the massive volume, variety and rapidly growing amount of data generated everyday. This data is not able to stored, processed and analyzed effectively by using traditional databases and tools.

FEATURE



- **Variety:** includes many types of data, such as Structured, Unstructured, Semi-structured data.
- **Volume:** is a massive amount of data generated from different sources.
- **Velocity:** data is created, transmitted, and processed with a high speed *(often in real time)*

BENEFITS

Data is a key factor for any business to understand its customers and provide better services for them. Therefore, companies should start implementing Big Data as soon as possible to improve their performance and competitiveness.

02. Big Data Concepts

Concepts

This section includes three key conceptual areas:

1. **Processing Concepts**
2. **Data Storage & System Types Concepts**
3. **Data Modeling Concept**

1. Processing Concepts

The **Processing Concepts** part introduces fundamental concepts for working with large datasets, such as parallel and distributed processing, batch vs. stream processing, and data integration methods like ETL and ELT.

2.Data Storage & System Types

The **Data Storage & System Types Concepts** part introduces key data storage models and system types used in modern data architectures, including OLTP, OLAP, Data Warehouse, Data Lake, and Lakehouse. It explains their purposes, characteristics, and when to use each system for analytics or operational workloads.

3.Data Modeling

The **Data Modeling** part introduces the main data modeling approaches used in Big Data, including Star Schema, Snowflake Schema, Data Vault, and Slowly Changing Dimensions (SCD).

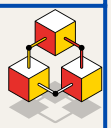
1. Processing Concepts

TYPE	DESCRIPTION
PARALLEL PROCESSING	**Parallel processing** is created to overcome hardware limits by splitting tasks into smaller pieces and running them at the same time using multiple threads or connected computers *(nodes)* in a single system *(cluster)*, in order to increase performance. 
DISTRIBUTED PROCESSING	**Distributed processing** is a method that connecting different computers to be a unified system to address mutual goals. **Structure:** • ***Worker node:*** is a normal node is used to calculate and process problem. ***Driver node:*** is a master node responsible for assigning tasks to worker nodes and coordinating the activities of the entire cluster. 
BATCH PROCESSING	**Batch processing** is created to handle with static data by splitting a large dataset into many smaller batches for parallel and distributed processing.
STREAM PROCESSING	**Stream processing** is created to process dynamic data in real time.
HYBRID PROCESSING	**Batch processing** + **Stream processing** to process data in real time or near real time.
ETL	**ETL** *(Extract, Transform, Load)* is a method of transferring data from multiple sources, transforming it into an acceptable format, and then loading it into the destination system. **Ex:** API, CRM,.. → Data Lake, Data Lake → Data Warehouse
ELT	**ELT** *(Extract, Load, Transform)* is a method of transferring data from sources into destination system, and then transforming data into acceptable format. **Ex:** Data Warehouse → Data Mart,...

2.Data Storage & System Types

TYPE	DESCRIPTION
DATA LAKE	**Data Lake** *(DL)* is a data storage designed to store large volumes of ***raw data*** in various formats, such as images, JSON, audio, videos, logs, and more. Data can be ingested from multiple sources, including APIs, databases, applications, or cloud services. For DS, DE **Characteristics**: • Stores massive amounts of raw data • Low cost • Highly scalable **Ex:** AWS S3, Azure Data Lake, Google Cloud Storage,... 
DATA WAREHOUSE	**Data Warehouse** *(DW)* is a data storage designed to store data for analytics and reporting. Data stored in a DW is always cleaned, transformed, and well-structured, making it ready for DA, DS, and business stakeholders such as Marketing or Sales teams. **Characteristics**: • Data is cleaned and transformed • Schema is clear • Typically designed with Star Schema or Snowflake Schema **Ex:** BigQuery, Redshift, Azure Synapse 
DATA LAKEHOUSE	**Data LakeHouse** is a combination between Data Lake and Data Warehouse. It stores raw data and transform. **Characteristics**: • Convenient **Ex**: Oracle, Data Lake + Trino / Delta Lake 
DATA MART	**Data Mart** is a data storage designed to serve a specific business domain or department. It usually receives data from a Data Warehouse. Data in DM is ready for reporting.
OLTP	**OLTP** *(Online Transaction Processing)* is a type of database system designed to handle real-time transactional operations for day-to-day business applications. 
OLAP	**OLAP** *(Online Analytics Processing)* is a type of database system designed to serve for reporting and analyzing. It usually contains denormalized table to support aggregation. 

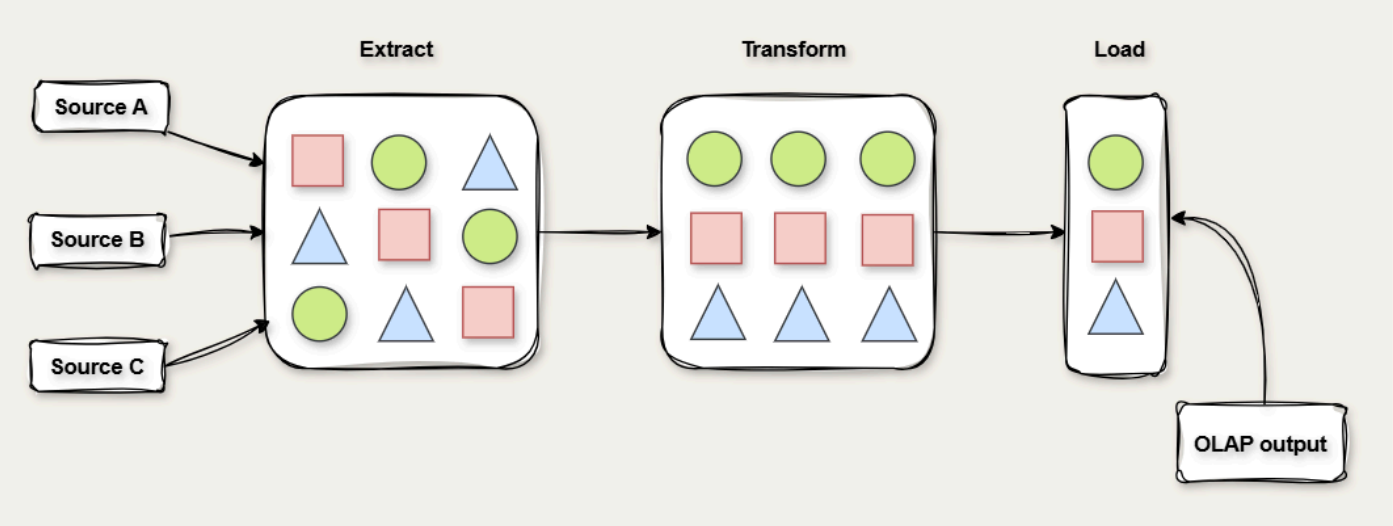
3.Data Modeling

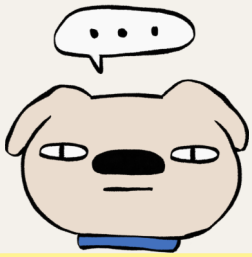
TYPE	DESCRIPTION
STAR SCHEMA	**Dimension + Fact** 
SNOWFLAKE SCHEMA	Extended Star Schema **Dimension + Fact + Subdimension**
FACT .VS DIM	**Dim**: • Dim is a table for describing feature of entities • Subdim is a table for supporting dim table **Ex**: *dim_customer, subdim_address, subdim_biography* **Fact** is a table for store information of transactions. 1. ***Transactional Fact*** store detailed of transactions. 2. ***Accumulative Fact*** store status of process. • **Ex**: *{ order_id: 1, date: 2025-10-29, shipping_status: True, received_status: False }* 3. ***Snapshot Fact*** store status of data in specific time • **Ex**: *{ date: 2025-10-29, total_revenue: 129.000.000đ }*
SLOWLY CHANGING DIMENSIONS (SCD)	SCD applies for dimension tables for tracking historical records.  • **SCD 1**: Overwrite old records • **SCD 2**: Store entire historical records (start/end date, version) • **SCD 3**: Store partial historical records (previous, current) • **SCD 6**: 1+2+3
DATA VAULT	**Hub + Link + Satellite**  • **Hub:** business keys (product_id, customer_id) • **Link**: Relationship among hubs. • **Satellite**: attributes Data vault is a complicated model designed to process multiple source and store detailed historical record. For Lakehouse.
NORMALIZATION	**Normalization** is a technique used to minimize data redundancy and ensure data integrity by organizing data into smaller, related tables. (OLTP)
DENORMALIZATION	**Denormalization** is a technique where normalized tables are combined or duplicated to reduce the number of joins and improve query performance. (OLAP)

03. Data Processing

Concepts

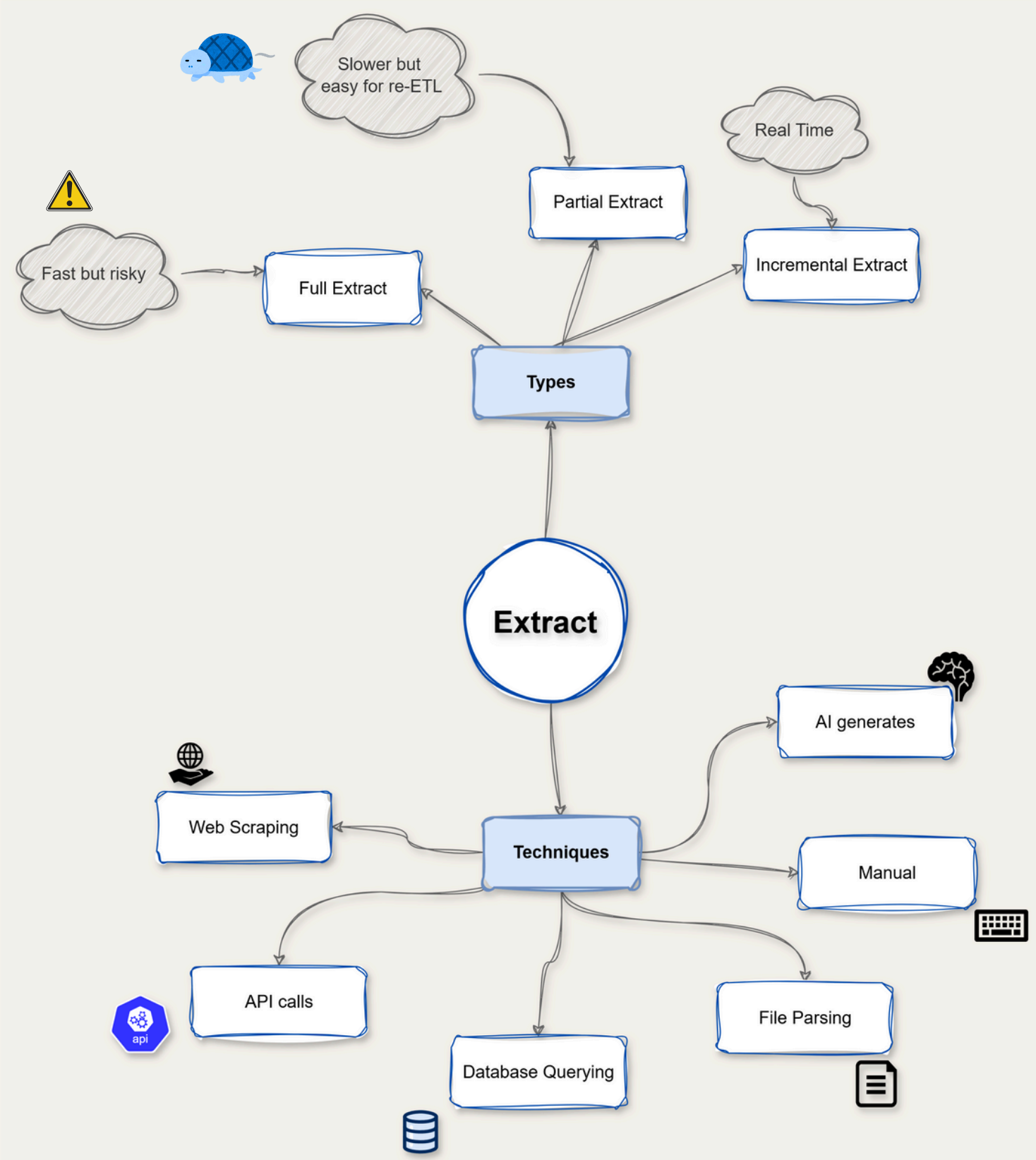
As mentioned in Section ***2.Big Data Concepts*** → ***1.Processing Concepts***, the purpose of the ETL process is to collect data from multiple sources, transform it into a clean and structured format, and load it into a centralized storage system for analysis.





HOW CAN WE ETL DATA FOR DEEPER ANALYSIS?

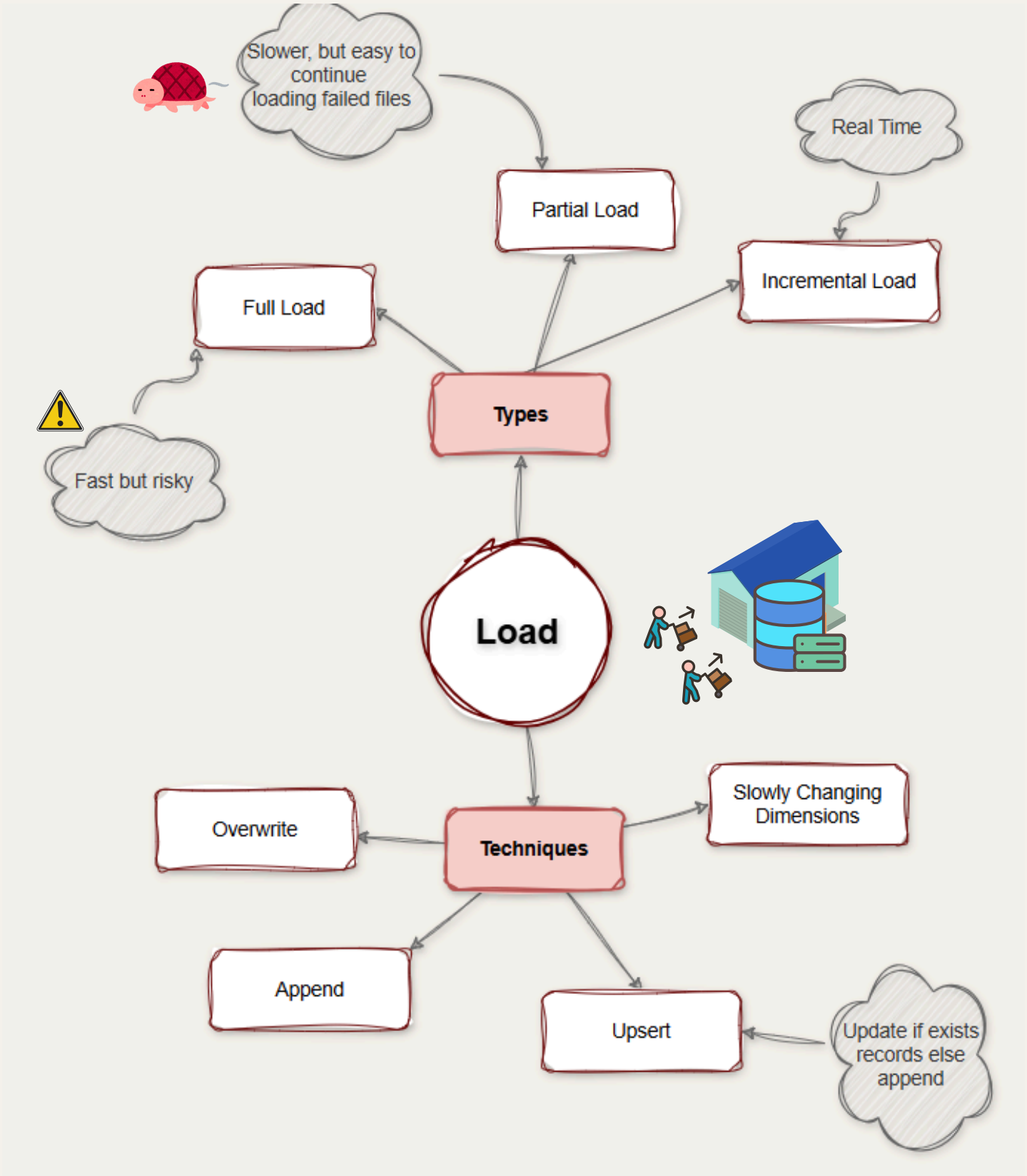
Extract


Slower but
easy for re-ETL
Real Time
Partial Extract
Fast but risky
Full Extract
Incremental Extract
Types
Extract
AI generates
Web Scraping
Techniques
Manual
API calls
File Parsing
Database Querying

Transform


Using AI for string data
Using RFM model
Add business rules & logic
Source A: Sex = M/F;
Source B: Sex = Male/Female
→ Standardize to Male/Female.
Regulation
Multiple sources
Data Aggregation
Data Enrichment
Data Standardlization
Data Integration
Data Calculation
Transform
Filter appropriate values
Ourlier Detection
Data Cleansing
Handle unwanted spaces
Drop Duplicates
Handle invalid values
Cast DataTypes
Handle NULL

Load


Slower, but easy to continue loading failed files
Real Time
Partial Load
Incremental Load
Full Load
Types
Fast but risky
Load
Overwrite
Techniques
Slowly Changing Dimensions
Append
Upsert
Update if exists records else append

04. Final Project

A. Introduction

Project Overview

This project demonstrates ETL to transform raw data into OLAP-ready outputs using techniques from **"03. Data Processing"**. ETL is performed in batch and streaming modes.

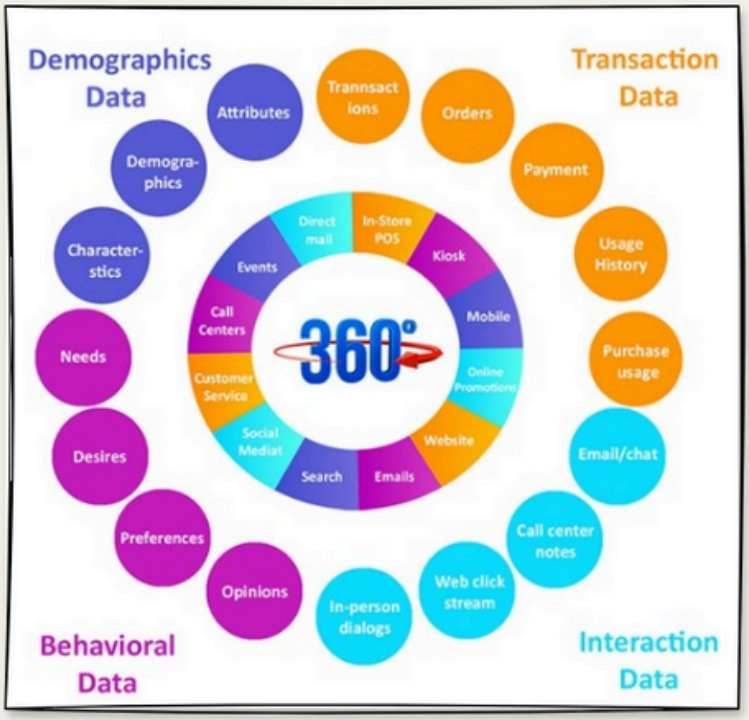
WHAT ARE THE INPUTS AND OUTPUTS OF FINAL PROJECT?

1. Batch ETL

Input: ***log-search*** and ***log-content*** folders (customer searches and contract interactions).
Output: OLAP output includes Data Warehouse and Data Mart for reporting and **Customer 360°** insights.

Customer 360

A **Customer 360°** view provides a unified and comprehensive understanding of each customer across all interactions and touchpoints. It enables businesses to personalize services, improve customer satisfaction, strengthen retention, and identify cross-sell or up-sell opportunities—ultimately driving better decision-making and increasing revenue.

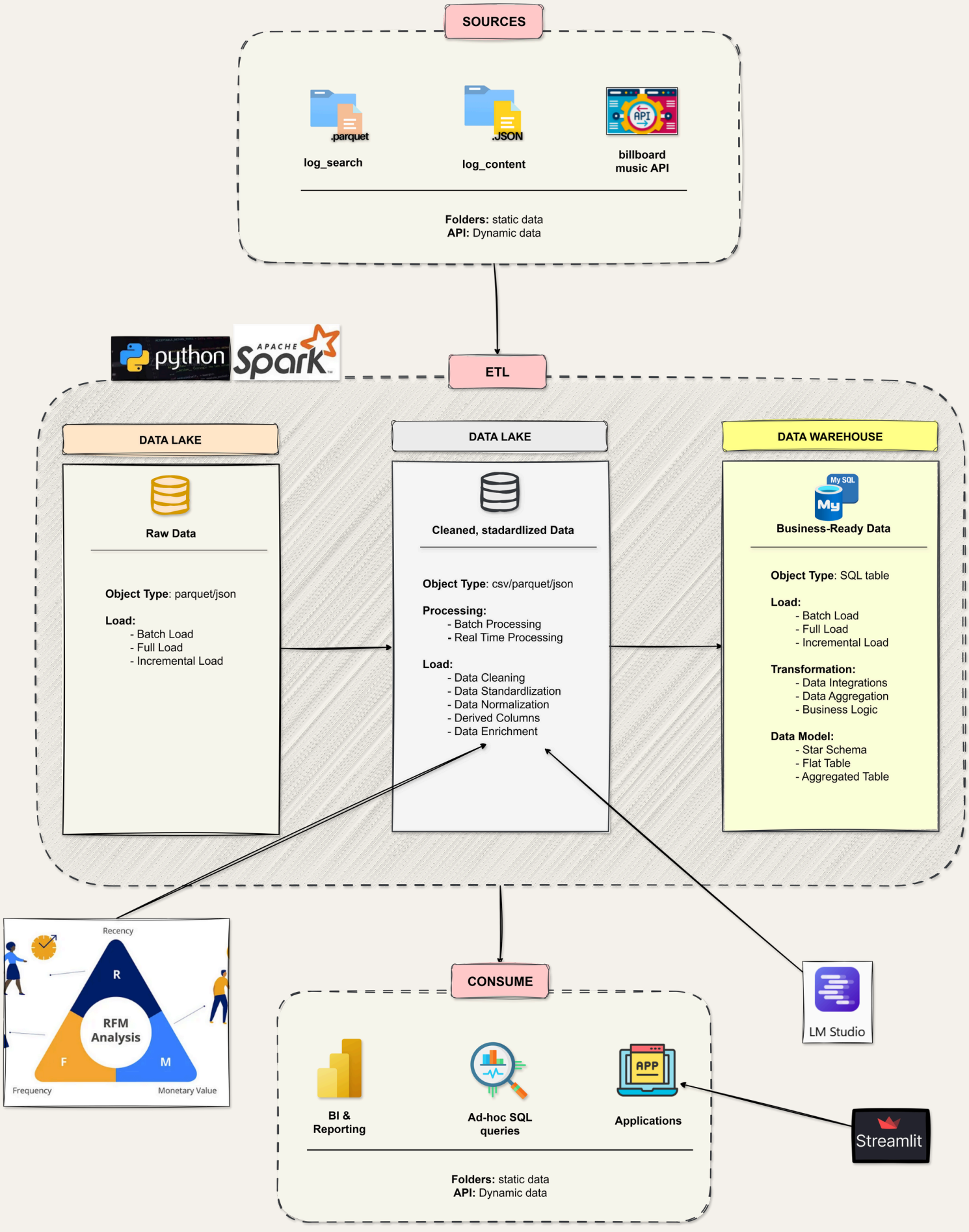


2. Streaming ETL

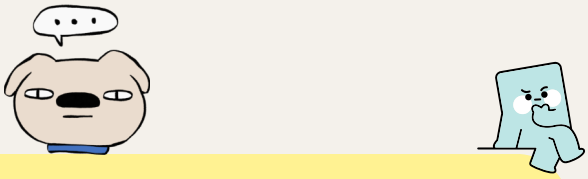
Input: Simulator API provided by the business, continuously sending song vote data in JSON format. Spark Streaming ingests this data in real-time, processing votes as they arrive.

Output: Small Streamlit app that displays a live votes leaderboard, showing which songs are leading based on real-time aggregated counts computed by Spark Streaming.

B. High Level Architecture



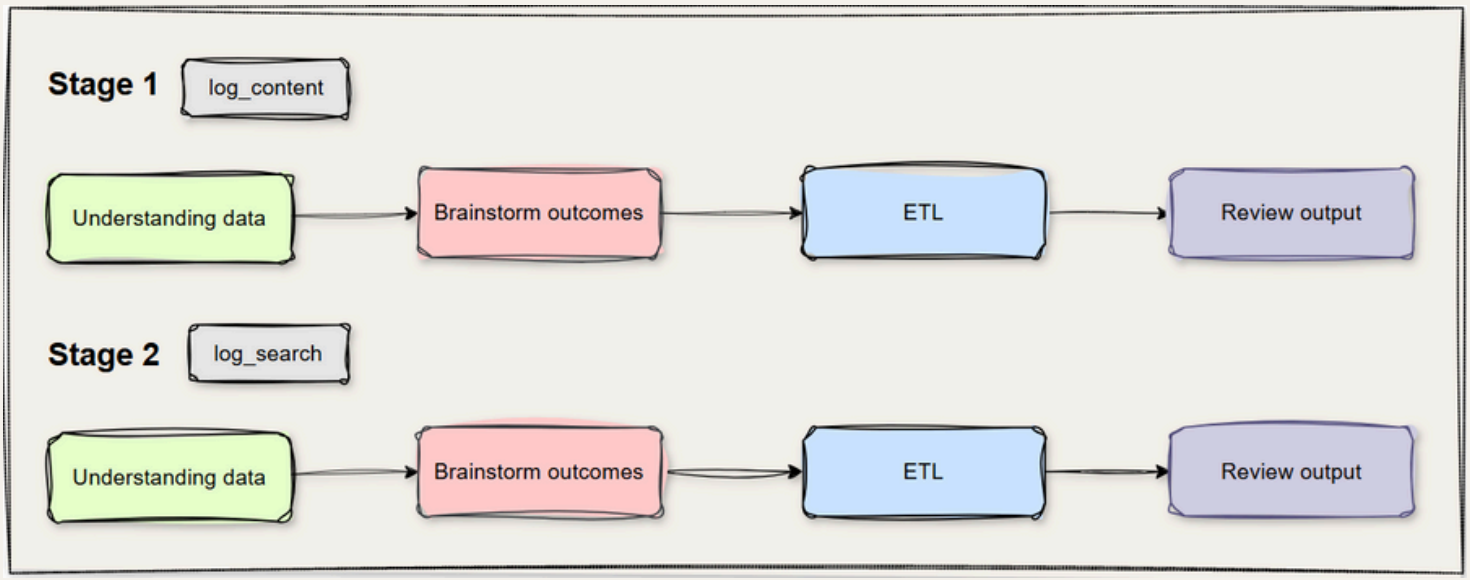
C. ETL Batch Data



WHERE TO START?

Before starting the ETL process, we will analyze the dataset to understand its structure, meaning, and business context and then brainstorm outcomes of the data warehouse. The diagram below outlines the steps involved. All source code is available in the following repository:
https://github.com/sonvuuu28/DE_2025_BigData_Project.git

Steps



Stage 1

Understanding log_content

Sample

AppName	Contract	Mac	TotalDuration
KPLUS	HNH579912	0C96E62FC55C	254
KPLUS	HUFD40665	CCEDDC333614	1457
KPLUS	HNH572635	B068E6A1C5F6	2318
KPLUS	HND141717	08674EE8D2C2	1452
KPLUS	HNH743103	402343C25D7D	251

Image: 5 sample records

AppName
KPLUS
RELAX
CHILD
CHANNEL
VOD
FIMS
SPORT
BHD
APP

Image: All values of "AppName"

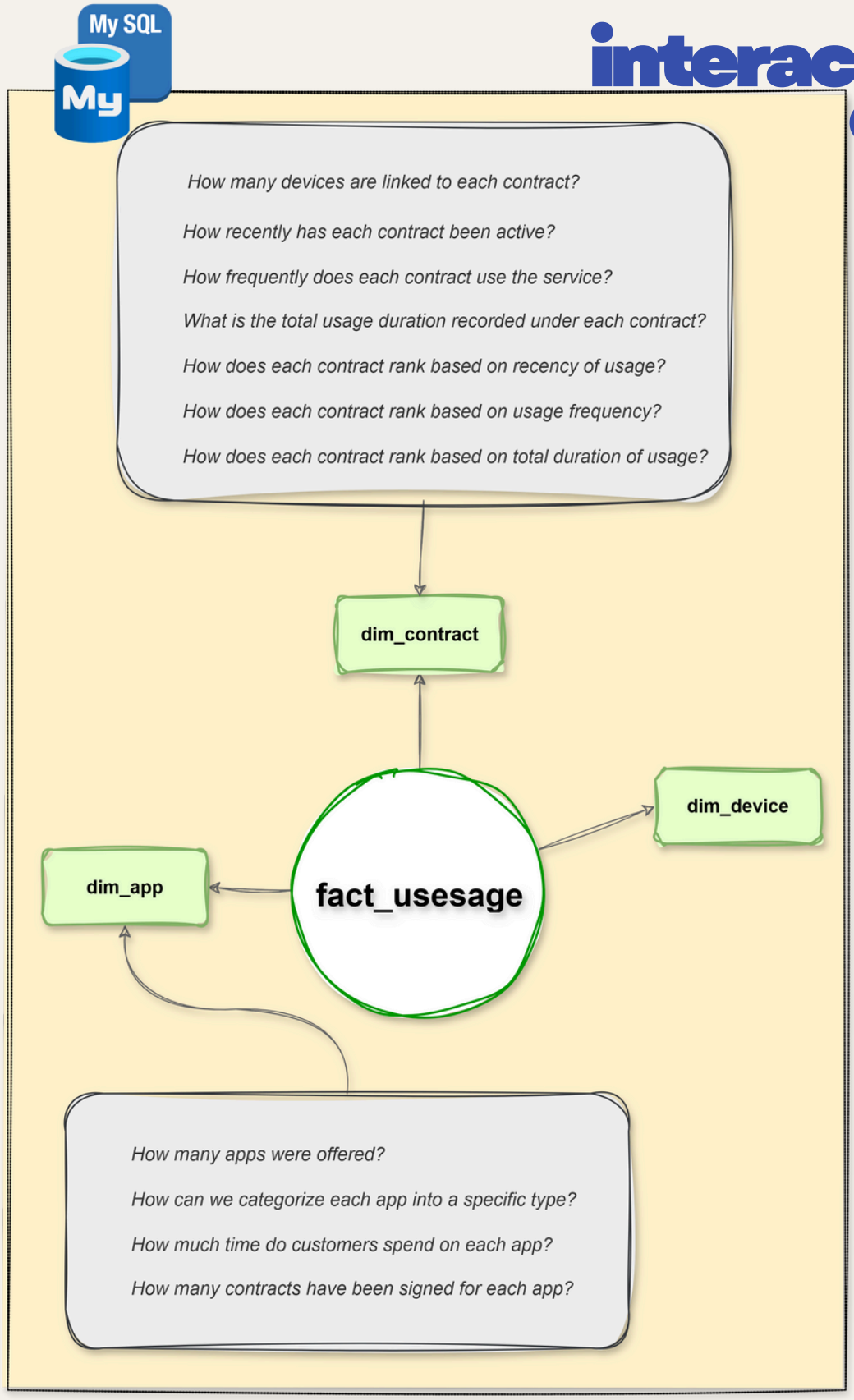
Description

COLUMNS	MEANINGS	DATA TYPE
App Name	The name of the application used by the customer.	String
Contract	The customer's contract ID	String
Mac	The MAC address of the device used to access the application	String
TOTAL DURATION	The total usage duration of the application for that contract and device.	Integer

Number of files:	30 files (from 01/04/2022 to 30/04/2022)
Type of files:	JSON
Numbers of records:	48.457.499

Brainstorm outcome of log_content

I will design the **log_content** star schema based on the architecture below and enrich each dimension table with additional metrics to answer the related business questions.



ETL log_content

Create database

```
CREATE DATABASE IF NOT EXISTS final_project;
USE final_project;
```

Initiation SparkSession

```
from pyspark.sql import SparkSession
from pyspark.sql.functions import *
from pyspark.sql.window import *
import os
from pyspark import StorageLevel
```

```
spark = SparkSession.builder.appName("final_project").config("spark.driver.memory", "8g").getOrCreate()
```

Process Data

```
def ensure_dir(path):
    """Tạo folder nếu chưa có"""
    if not os.path.exists(path):
        os.makedirs(path)


def read_json(path_name):
    df = spark.read.json(path_name)
    df = df.select("_source.*")
    df = df.withColumnRenamed("AppName", "app_name")
    df = df.withColumnRenamed("Contract", "contract")
    df = df.withColumnRenamed("Mac", "mac")
    df = df.withColumnRenamed("TotalDuration", "total_duration")

    event_date = path_name.split("\\")[-1].split(".")[0]
    df = df.withColumn("event_date", to_date(lit(event_date), "yyyyMMdd"))
    return df


def add_id(df, name):
    df = df.withColumn(name, monotonically_increasing_id())
    return df
```

Rename column with snake rule

add date by name of files

add Surrogate Key for dimension table

Process Data

Add category column for understanding the meaning of App name

```
def categorize_app(df):
    df = df.withColumn(
        "category",
        when((col("app_name") ==  "KPLUS") | (col("app_name") ==  "RELAX") | (col("app_name") ==  "CHILD")
             | (col("app_name") ==  "CHANNEL") | (col("app_name") ==  "SPORT")
        , "Kênh truyền hình")
        .when((col("app_name") ==  "VOD") | (col("app_name") ==  "FIMS") | (col("app_name") ==  "BHD")
        , "Phim và nội dung theo yêu cầu")
        .when((col("app_name") ==  "Apps"), "Ứng dụng")
    )

    return df


def add_metrics_contract(df):
    df = df.groupBy("contract").agg(
        sum("total_duration").alias("contract_total_duration"),
        count_distinct("mac").alias("total_device"),
        date_diff(to_date(lit("2022-04-30"), 'yyyy-MM-dd'), max("event_date")).alias("recency"),
        count_distinct("event_date").alias("frequency")
    )
    return df


def add_quartiles(df, quantile_columns):
    cols = list(quantile_columns.keys())
    quantiles_list = df.approxQuantile(cols, [0.25, 0.5, 0.75], 0.1)

    for idx, col_name in enumerate(cols):
        q1, q2, q3 = quantiles_list[idx]
        score_col = quantile_columns[col_name]

        if col_name == "recency":
            df = df.withColumn(
                score_col,
                when(col(col_name) <= q1, 4)
                .when((col(col_name) > q1) & (col(col_name) <= q2), 3)
                .when((col(col_name) > q2) & (col(col_name) <= q3), 2)
                .otherwise(1),
            )
        else:
            df = df.withColumn(
                score_col,
                when(col(col_name) <= q1, 1)
                .when((col(col_name) > q1) & (col(col_name) <= q2), 2)
                .when((col(col_name) > q2) & (col(col_name) <= q3), 3)
                .otherwise(4),
            )
    return df
```

Enrich data by add business metrics and RFM metrics

Add quartile column for categorizing customer classes

Create Star Schema Model

```python
def create_dim_app(df):
    df = df.select("app_name", "contract", "total_duration")
    df = df.groupBy("app_name").agg(sum("total_duration").alias("app_total_duration"), count_distinct("contract").alias("app_total_active_contracts"))
    df = categorize_app(df)
    df = add_id(df, "app_id")
    df = df.select("app_id", "category", "app_name", "app_total_duration", "app_total_active_contracts")
    return df

def create_dim_contract(df):
    df = add_metrics_contract(df)
    quantile_cols = {
        "recency": "recency_score",
        "frequency": "frequency_score",
        "contract_total_duration": "duration_score"
    }

    df = add_quartiles(df, quantile_cols)

    df = add_id(df, "contract_id")
    df = df.select("contract_id", "contract", "total_device", "recency", "frequency", "contract_total_duration", "recency_score", "frequency_score", "duration_score")
    return df

def create_dim_device(df):
    df = df.select("mac")
    df = df.dropDuplicates()
    df = add_id(df, "device_id")
    df = df.select("device_id", "mac")
    return df

def create_fact_usage(df, dim_app, dim_contract, dim_device):
    df = df.join(broadcast(dim_app), on=["app_name"],how="left")
    df = df.join(dim_contract, on=["contract"],how="left")
    df = df.join(dim_device, on=["mac"],how="left")

    df = df.select("app_id", "contract_id", "device_id", "total_duration", "event_date")
    return df
```

Read files

I chose the batch extract method, where each file is read individually and then unioned into a single dataset.

```python
def read_all_files():
    project_path = os.getcwd()
    log_content_path = os.path.join(project_path, "log_content")
    list_files = os.listdir(log_content_path)

    union_df = None
    for file_name in list_files:
        path_name = os.path.join("./log_content", file_name)
        df = read_json(path_name)

        if union_df is None:
            union_df = df
        else:
            union_df = union_df.unionByName(df, allowMissingColumns=True)

    return union_df
```

Read files

write_to_disk saves the DataFrame as Parquet to local storage, while **write_to_mysql** writes the DataFrame to a MySQL table via JDBC (default mode: overwrite).

```python
def write_to_disk(df, name):
    base_path = "./data_content"
    ensure_dir(base_path)
    output_path = os.path.join(base_path, name)
    ensure_dir(output_path)
    df.write.mode("overwrite").parquet(output_path)


def write_to_mysql(df, table_name, mode="overwrite"):
    url = "jdbc:mysql://localhost:3306/final_project"
    props = {"user": "root", "password": "12345", "driver": "com.mysql.cj.jdbc.Driver"}
    df.write.jdbc(url=url, table=table_name, mode=mode, properties=props)
```

Main

```python
def main():
    df = read_all_files().persist(StorageLevel.MEMORY_AND_DISK)

    dim_app = create_dim_app(df)
    dim_contract = create_dim_contract(df).persist(StorageLevel.MEMORY_AND_DISK)
    dim_device = create_dim_device(df)

    fact_usage = create_fact_usage(df, dim_app, dim_contract, dim_device).persist(
        StorageLevel.MEMORY_AND_DISK
    )

    # Save to Parquet
    write_to_disk(dim_app, "dim_app")
    write_to_disk(dim_contract, "dim_contract")
    write_to_disk(dim_device, "dim_device")
    write_to_disk(fact_usage, "fact_usage")

    # Save to MySQL
    write_to_mysql(dim_app, "dim_app", "overwrite")
    write_to_mysql(dim_contract, "dim_contract", "overwrite")
    write_to_mysql(dim_device, "dim_device", "overwrite")
    write_to_mysql(fact_usage, "fact_usage", "append")

    df.unpersist()
    dim_contract.unpersist()
    fact_usage.unpersist()
    print("✅ Data warehouse written successfully.")


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

Review output log_content

dim_app

app_id	category	app_name	app_total_duration	app_total_active_contracts
0	Kênh truyền hình	KPLUS	544236032	35803
1	Kênh truyền hình	RELAX	44568537	19027
2	Kênh truyền hình	CHILD	395666651	53987
3	Phim và nội dung theo yêu cầu	FIMS	14453227	1620
4	Kênh truyền hình	CHANNEL	48113594728	1497345
5	Kênh truyền hình	SPORT	3470274	4174
6	Phim và nội dung theo yêu cầu	VOD	1266490639	124098

dim_contract

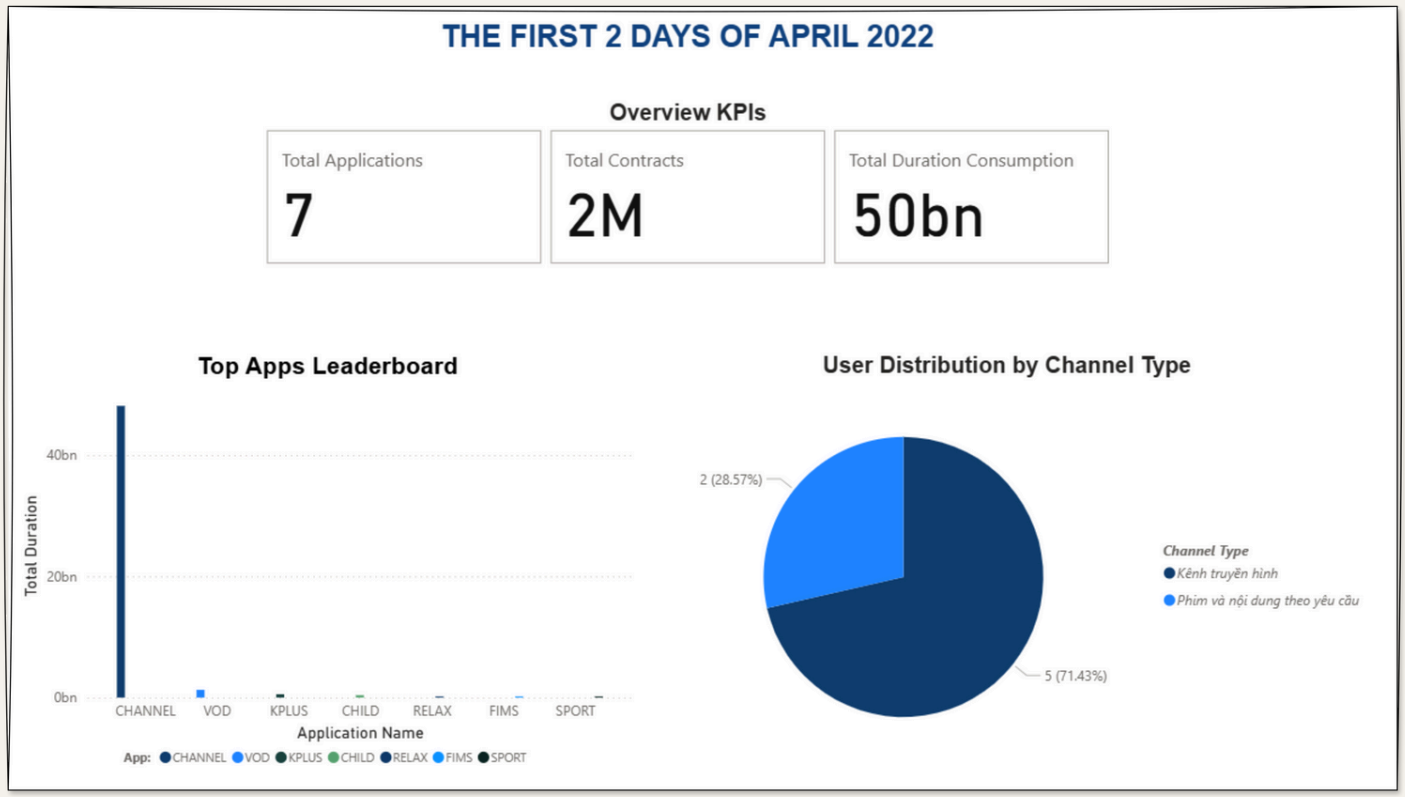
contract_id	contract	total_device	recency	frequency	contract_total_duration	recency_score	frequency_score	duration_score
42949672960	TIFD09369	1	28	2	169104	4	1	4
0	DAFD47733	2	28	2	30909	4	1	3
85899345920	BND020123	1	28	2	33774	4	1	4
77309411328	NBFD17791	1	28	2	52	4	1	1
51539607552	HPFD88431	1	28	2	99136	4	1	4
25769803776	BNFD58346	1	28	2	8184	4	1	2
85899345921	HNH592681	1	28	2	54471	4	1	4
51539607553	TQFD07672	1	28	2	19635	4	1	3
1	NTFD33896	3	28	2	56907	4	1	4

dim_device

device_id	mac
0	B068E6A1C5F6
85899345920	10394E27A69F
1	CCD4A1FA86A5
85899345921	8CC84B700F19
2	B84DEED34AA5
3	08674EE196F6
85899345922	D89C67A90765
85899345923	4CEBBDD6C08F

fact_usage

app_id	contract_id	device_id	total_duration	event_date
3	1288490191093	94489287424	50460	2022-04-01
4	1640677510962	1149	3335	2022-04-01
3	584115553910	60129544109	16	2022-04-01
3	300647710721	85899353787	7209	2022-04-02
3	1477468753891	17179872877	15062	2022-04-01
3	77309415059	34359739249	79248	2022-04-01
3	687194769894	60129545122	620	2022-04-01

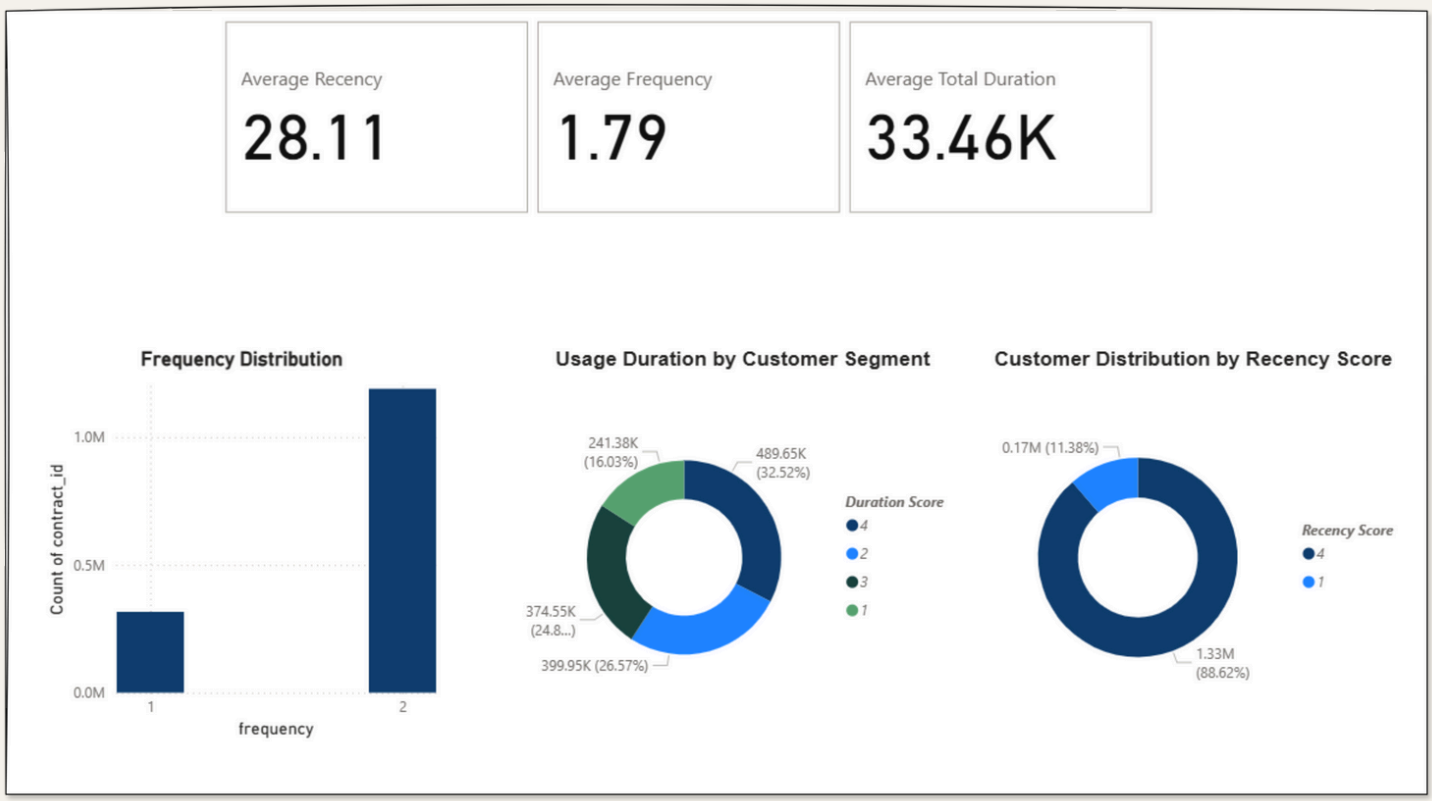


QUICK REVIEW

KPIs: Excellent overall performance with 2M Contracts using services and 50bn Total Duration Consumption achieved in just two days.

Heavy Reliance on Channels: The business is heavily dominated by CHANNEL consumption

Major Gap: VOD (Films and On-demand content) usage is weak. It only gets 28.57% of the user activity. In comparison, Live TV Channels are dominating with 71.43%.



QUICK REVIEW

Key Metrics
- Average Recency: 28.1 days → Customers return rarely
- Average Frequency: 1.79 → Very low usage

1. Frequency Distribution
- Most customers (>1.0M contracts) use the service twice
- Around 0.4M contracts have a frequency of 1
- No customers have frequency ≥ 3, confirming very low engagement

2. Usage Duration by Customer Segment
- Customers with Duration Score 4 contribute most to total usage (32.52%, 489.65K)
- Next largest segment is Duration Score 2 (26.57%, 399.95K)
- Indicates a small group of heavy users dominates usage

3. Customer Distribution by Recency Score
- Majority of customers have Recency Score 1 (88.62%, 1.33M contracts)
- Only a small portion have Recency Score 4 (11.38%, 0.17M contracts)
- High proportion of Recency Score 1 indicates high churn risk

STAGE 2

Understanding log_search

Sample

user_id	keyword
NULL	trữ tình
44887906	trữ tình
2719170	bolero
NULL	amy schumer: trực tiếp từ nhà hát apollo
8830996	cậu mang à sĩ hanako
NULL	Hoa trong bao
41559909	liên minh công lý phiên bản của zack snyder (đen trắng)
92715770	NULL
49026196	việt nam vs appa
NULL	chuyển sinh thành nhện
41376437	nhất kiến khuynh tâm
1254139	giác
42534799	nexsport
49190631	Tìm kiếm bằng giọng nói
NULL	Lương Sơn Bá Chúc Anh Đài

Image: sample records

Description

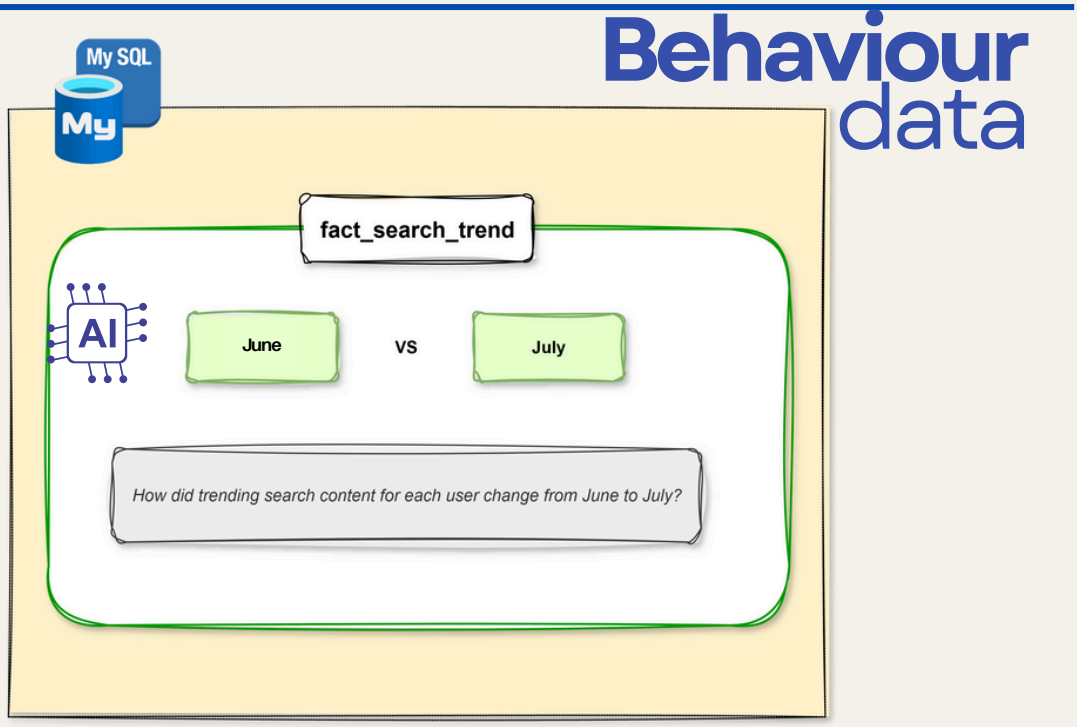
COLUMNS	MEANINGS	DATA TYPE
user_id	id of user	String
keyword	the search term entered by the user	String

60 Folders from 01/06/2022 to 31/07/2022

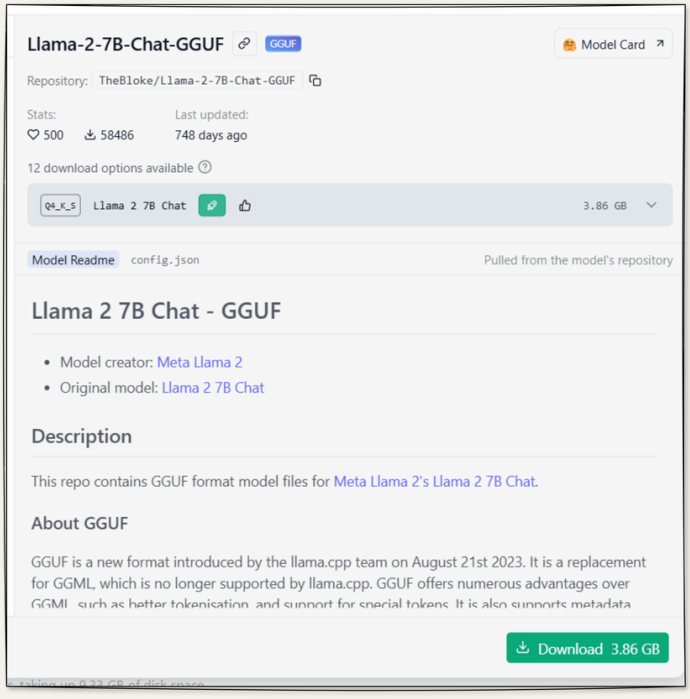
Brainstorm outcome of log_search

I will design a flat **log_search** table and apply an AI model to classify user search intent, enabling analysis of how search behavior changed between June and July.

Since search data is text, an AI-based classification model is required to categorize it effectively.



I used the **Llama-2-7B-Chat-GGUF** model to classify the search inputs. The model was trained with **7 billion parameters**, which provides sufficient capability and accuracy for my use case.



ETL log_search

Process Data

```python
def read_files(month):
    folders = os.listdir("./log_search/")
    df = None
    for folder in folders:
        if int(folder[4:6]) == month:
            if df is None:
                df = spark.read.parquet(os.path.join("./log_search", folder))
            else:
                df_next = spark.read.parquet(os.path.join("./log_search", folder))
                df = df.union(df_next)

    return df


def filter_most_search(month):
    df = read_files(month)
    df = df.select("user_id", "keyword")
    df = df.groupBy("user_id", "keyword").count()
    win_spec = Window.partitionBy("user_id").orderBy(col("count").desc())
    df = df.withColumn("rank", row_number().over(win_spec))
    df = df.filter(col("rank") == 1)
    df = df.withColumnRenamed("keyword", "most_search")
    df = df.select("user_id", "most_search")
    return df


def write_to_disk(df, month):
    path = f"./output/t{month}"
    if not os.path.exists(path):
        os.makedirs(path)
    df.write.mode("overwrite").option("header", "True").csv(path)
```

read files June and July

Remove records with null users and extract only the top search queries.

Write dataframe to store top search of users

```python
def main():
    t6 = etl(6)
    t7 = etl(7)
    write_to_disk(t6, 6)
    write_to_disk(t7, 7)
    print("✅ Written successfully.")


main()
```

Temporarily write the t6 and t7 DataFrames to storage so they can be processed by the AI model for classification.

Classify category

Prompt

```
system_prompt = """
Bạn là chuyên gia phân loại nội dung phim, chương trình truyền hình và các loại nội dung giải trí.
Bạn sẽ nhận một danh sách tên có thể viết sai, viết liền không dấu, viết tắt, hoặc chỉ là cụm từ liên quan đến nội dung.

Nguyên tắc:
- Không được trả về "Other" nếu có thể đoán được dù chỉ một phần ý nghĩa.
- Luôn cố gắng sửa lỗi, nhận diện tên gần đúng hoặc đoán thể loại gần đúng.
- Nếu không chắc → chọn thể loại gần nhất (VD: mô tả tình cảm → Romance, thể thao → Sports, v.v.)

Nhiệm vụ của bạn:
1. **Chuẩn hoá tên**: thêm dấu tiếng Việt nếu cần, tách từ, chỉnh chính tả (vd: "thuyetminh" → "Thuyết minh", "tramnamu" → "Trăm năm hữu duyên", "capdoi" → "Cặp đôi").
2. **Nhận diện tên hoặc ý nghĩa gốc gần đúng nhất**. Bao gồm:
- Tên phim, series, show, chương trình
- Quốc gia / đội tuyển (→ "Sports" hoặc "News")
- Từ khoá mô tả nội dung (→ phân loại theo ý nghĩa, ví dụ "thuyếtminh" → "Other" hoặc "Drama", "bigfoot" → "Horror")
3. **Gán thể loại phù hợp nhất** trong các nhóm sau:
- Action
- Romance
- Comedy
- Horror
- Animation
- Drama
- C Drama
- K Drama
- Sports
- Music
- Reality Show
- TV Channel
- News
- Other

Một số quy tắc gợi ý nhanh:
- Có từ "VTV", "HTV", "Channel" → TV Channel
- Có "running", "master key", "reality" → Reality Show
- Quốc gia, CLB bóng đá, sự kiện thể thao → Sports hoặc News
- "sex", "romantic", "love" → Romance
- "potter", "hogwarts" → Drama / Fantasy
- Tên phim Việt/Trung/Hàn → ưu tiên Drama / C Drama / K Drama

Hãy chỉ trả về **một JSON hợp lệ duy nhất**, không kèm giải thích, không có markdown, không có gạch đầu dòng.

Ví dụ:
{
  "cẩm tú nam ca": "Romance",
  "killing eve": "Drama",
  "fairy tail": "Animation"
}
"""
user_prompt = f"Danh sách: {movie_list}"
```

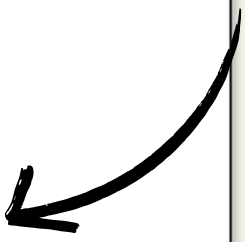
```
try:
    response = requests.post(
        "http://127.0.0.1:1234/v1/chat/completions",
        headers={"Content-Type": "application/json"},
        json={
            "model": "llama-3-groq-8b-tool-use",
            "messages": [
                {"role": "system", "content": system_prompt},
                {"role": "user", "content": user_prompt},
            ],
            "temperature": 0,
            # "max_tokens": 1024
        },
        timeout=120,
    )

    result = response.json()["choices"][0]["message"]["content"].strip()
    result = json.loads(result)
    print(result)

    return result

except Exception as e:
    print("❌ Error:", e)
    return {m: "Other" for m in movie_list}
```

Call AI model

ETL Data

```python
def preprocess(t6, t7):
    t6 = t6.withColumnRenamed("most_search", "most_search_t6").withColumnRenamed(
        "category", "category_t6"
    )

    t7 = t7.withColumnRenamed("most_search", "most_search_t7").withColumnRenamed(
        "category", "category_t7"
    )

    return t6, t7


def analyze_trending(t6, t7):
    df = t6.join(t7, on="user_id", how="outer")

    df = df.withColumn(
        "trending_type",
        when(col("category_t6") == col("category_t7"), "Unchanged").otherwise(
            "Changed"
        ),
    )

    df = df.withColumn(
        "previous",
        when(col("trending_type") == "Unchanged", "Unchanged").otherwise(
            concat_ws("->", col("category_t6"), col("category_t7"))
        ),
    )
    return df


def write_to_mysql(df, table_name, mode="overwrite"):
    url = "jdbc:mysql://localhost:3306/final_project"
    props = {"user": "root", "password": "12345", "driver": "com.mysql.cj.jdbc.Driver"}
    df.write.jdbc(url=url, table=table_name, mode=mode, properties=props)


def main():
    t6, t7 = load_data()
    t6, t7 = preprocess(t6, t7)
    df_final = analyze_trending(t6, t7)

    write_to_mysql(df_final, "fact_trending_change", "overwrite")


main()
```

Compare 2 month



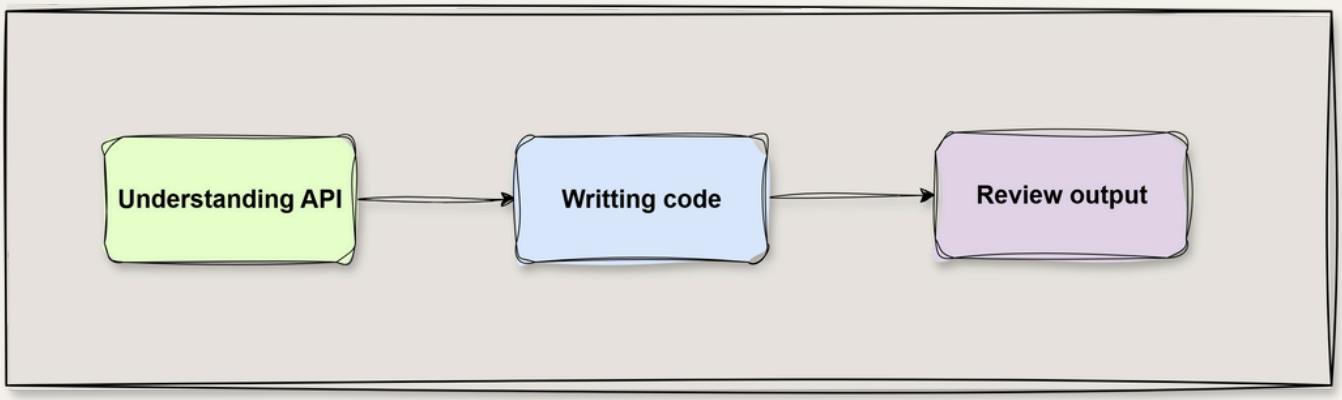
Review output log_content

fact_trending_change

user_id	most_search_t6	category_t6	most_search_t7	category_t7	trending_type	previous
98630244	quý ông hoàn hảo và cô nàng tạm được	Comedy	điên thì có sao	Horror	Changed	Comedy->Horror
98631046	minh lan truyện	C Drama	pachinko	Other	Changed	C Drama->Other
98632118	tung hoành	Sports	termi	Horror	Changed	Sports->Horror
98636848	Bắt Ma Phá Án	Horror	hoa du kí	Drama	Changed	Horror->Drama
98653848	mẹ chồng	Drama	bí kíp luyện rồng	Other	Changed	Drama->Other
98655119	món quà cầu vồng	Romance	my band	Music	Changed	Romance->Music
98655593	produce 48	Reality Show	hoàn hồn	Comedy	Changed	Reality Show->Comedy
98660487	KAROKO	Music	sakura - thủ lĩnh thẻ bài	Animation	Changed	Music->Animation
98660797	đừng làm phiền tôi học tập	Comedy	cuộc chiến hậu cung	Action	Changed	Comedy->Action

D. ETL Stream Data

Architecture



Understanding API

FIELD	DESCRIPTION	TYPE
STATUS	Status of the request ("ok" if successful)	String
COUNT	Number of votes returned in this response	Integer
DELAY_USED	Actual delay used in seconds before sending the response	Float
VOTES	Array containing individual vote objects	Array[Object]

Inside each vote object

USER_ID	Unique user identifier (UUID format)	String
SONG	Title of the song and the artist	String
VOTE	Indicates if the user voted (true)	Boolean
TIMESTAMP	timestamp of when the vote occurred	String

Code

Structure

- API.py
- get_data.py
- spark_stream.py
- streamlit.py

API.py

Generates one vote.

Returns a batch of random votes.

API info/documentation.

```python
def generate_vote():
    return {
        "user_id": str(uuid.uuid4()),
        "song": random.choice(SONGS),
        "vote": True,
        "timestamp": datetime.now().isoformat(),
    }


# ========================
# API GET /votes
# Trả về danh sách bình chọn ngẫu nhiên
# ========================
@app.route("/votes", methods=["GET"])
def get_votes():
    # Nếu không truyền count → random 1-20
    count = request.args.get("count")
    count = int(count) if count is not None else random.randint(20, 80)

    # Nếu không truyền delay → random 0-2 giây
    delay = request.args.get("delay")
    delay = float(delay) if delay is not None else random.uniform(0, 2)

    time.sleep(delay)  # mô phỏng server chậm

    votes = [generate_vote() for _ in range(count)]

    # Trả về JSON thủ công (đảm bảo không bị encode Unicode)
    return app.response_class(
        response=json.dumps(
            {
                "status": "ok",
                "count": len(votes),
                "delay_used": round(delay, 3),
                "votes": votes,
            },
            ensure_ascii=False,
        ),
        status=200,
        mimetype="application/json",
    )


# Trang chủ để giới thiệu API
@app.route("/", methods=["GET"])
def info():
    return jsonify(
        {
            "service": "billboard-music-api",
            "description": "API mô phỏng bình chọn bài hát US-UK.",
            "endpoints": {
                "/votes": "GET → Trả về danh sách bình chọn ngẫu nhiên",
            },
            "example": "http://localhost:1111/votes?count=5&delay=1",
        }
```

Code

get_data.py

Continuously fetch votes from simulator API and store them as JSON.

```python
API_URL = "http://localhost:1111/votes"
OUTPUT_DIR = "raw_data"

os.makedirs(OUTPUT_DIR, exist_ok=True)

while True:
    try:
        res = requests.get(API_URL)
        data = res.json()

        timestamp = datetime.utcnow().strftime("%Y%m%d_%H%M%S_%f")
        file_path = f"{OUTPUT_DIR}/votes_{timestamp}.json"

        with open(file_path, "w", encoding="utf-8") as f:
            json.dump(data, f, ensure_ascii=False)

        print(f"✅ Saved: {file_path} ({data['count']} votes)")
    except Exception as e:
        print("❌ Error:", e)

    time.sleep(10)  # poll mỗi 2 giây
```

Code

spark_stream.py

```python
# ===== 7. Hàm ghi batch =====
def save_snapshot(df, batch_id):
    # Ghi overwrite → snapshot cuối cùng luôn cập nhật
    df.write.mode("overwrite").csv(OUTPUT_DIR)
    # In console an toàn Windows
    print(f"Batch {batch_id} written to {OUTPUT_DIR}")
    # In realtime tổng votes
    df.show(truncate=False)
```

Overwrite CSV → always maintain the latest snapshot.

```python
# ===== 8. Start streaming =====
query = (
    song_counts.writeStream.outputMode("complete")  # complete để tổng votes
    .foreachBatch(save_snapshot)  # ghi CSV + show console
    .trigger(processingTime="10 seconds")  # đọc file mới mỗi 10s
    .start()
)
```

Start Streaming

```python
# ===== 9. Chờ stop =====
query.awaitTermination()
```

keep the stream running continuously.

Code

Streamlit

streamlit.py

Launches Spark streaming script (stream.py) in a new console.

```python
def run_spark_stream():
    subprocess.Popen(
        f'"{SPARK_SUBMIT}" --master local[2] stream.py',
        shell=True,
        creationflags=subprocess.CREATE_NEW_CONSOLE,
    )


if st.button("Start API"):
    subprocess.Popen(["python", "api.py"])
    st.success("✅ API Started")

if st.button("Save Vote Raw"):
    subprocess.Popen(["python", "get_data.py"])
    st.success("📌 Poller Started – generating votes...")

if st.button("Start Spark Stream"):
    cmd = 'start cmd /k "spark-submit --master local[2] spark_stream.py"'
    subprocess.Popen(cmd, shell=True)
    st.success("⚡ Spark Streaming Started (check new CMD window)")


if st.button("Stop All"):
    for proc in psutil.process_iter():
        try:
            cmd = " ".join(proc.cmdline())
            if (
                "api.py" in cmd
                or "get_data.py" in cmd
                or "spark_stream.py" in cmd
                or "spark-submit" in cmd
            ):
                proc.kill()
        except:
            pass
    st.warning("🛑 All jobs stopped")
```

Launches api.py via Python subprocess.

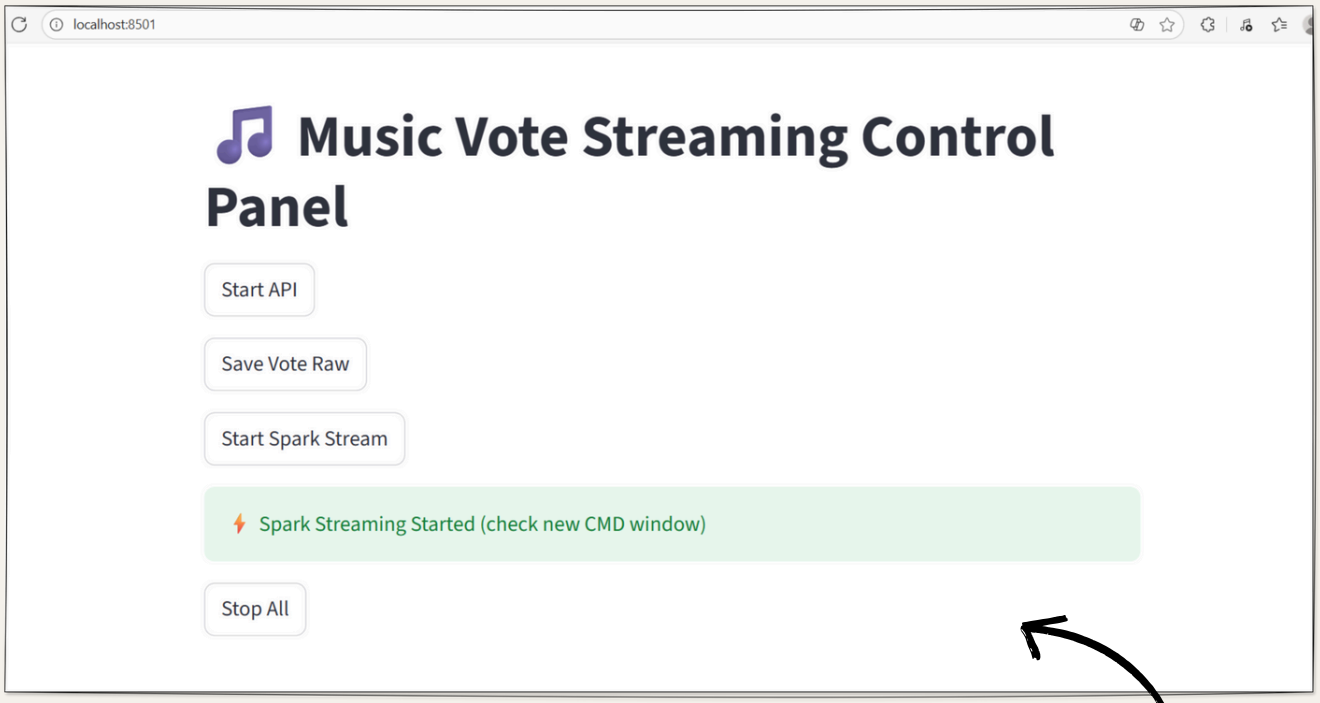
Runs get_data.py

Executes spark_stream.py using spark-submit

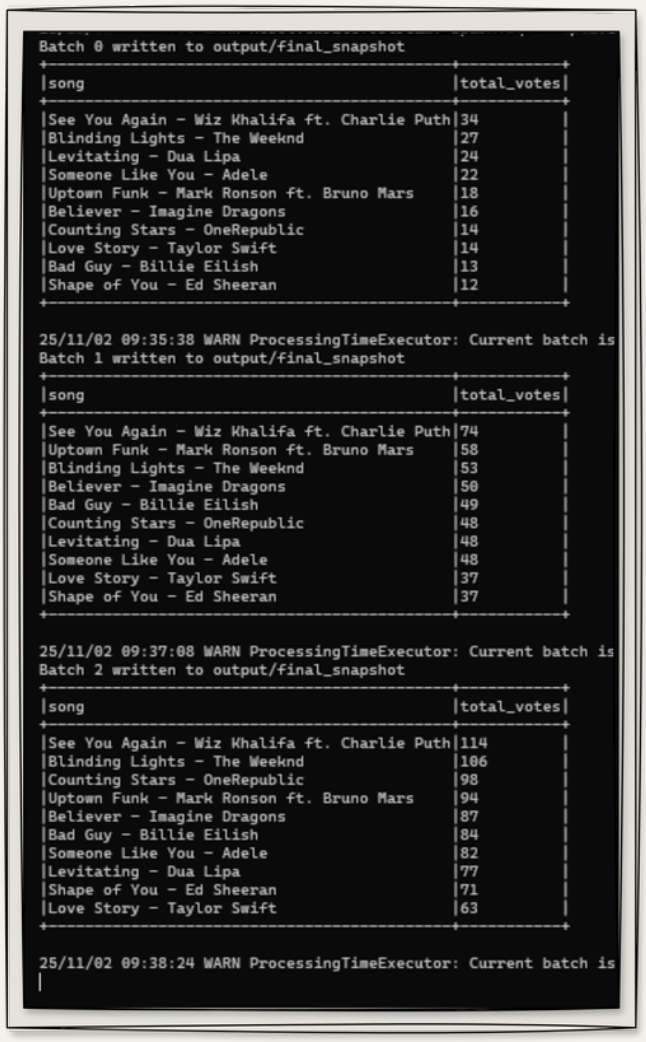
Stop all

Output

interface



Click respectively button to execute codes.



```
Batch 0 written to output/final_snapshot
+----------------------------------------------+-----------+
|song                                          |total_votes|
+----------------------------------------------+-----------+
|See You Again - Wiz Khalifa ft. Charlie Puth|34         |
|Blinding Lights - The Weeknd                  |27         |
|Levitating - Dua Lipa                         |24         |
|Someone Like You - Adele                      |22         |
|Uptown Funk - Mark Ronson ft. Bruno Mars      |18         |
|Believer - Imagine Dragons                    |16         |
|Counting Stars - OneRepublic                  |14         |
|Love Story - Taylor Swift                     |14         |
|Bad Guy - Billie Eilish                       |13         |
|Shape of You - Ed Sheeran                     |12         |
+----------------------------------------------+-----------+

25/11/02 09:35:38 WARN ProcessingTimeExecutor: Current batch is
Batch 1 written to output/final_snapshot
+----------------------------------------------+-----------+
|song                                          |total_votes|
+----------------------------------------------+-----------+
|See You Again - Wiz Khalifa ft. Charlie Puth|74         |
|Uptown Funk - Mark Ronson ft. Bruno Mars      |58         |
|Blinding Lights - The Weeknd                  |53         |
|Believer - Imagine Dragons                    |50         |
|Bad Guy - Billie Eilish                       |49         |
|Counting Stars - OneRepublic                  |48         |
|Levitating - Dua Lipa                         |48         |
|Someone Like You - Adele                      |48         |
|Love Story - Taylor Swift                     |37         |
|Shape of You - Ed Sheeran                     |37         |
+----------------------------------------------+-----------+

25/11/02 09:37:08 WARN ProcessingTimeExecutor: Current batch is
Batch 2 written to output/final_snapshot
+----------------------------------------------+-----------+
|song                                          |total_votes|
+----------------------------------------------+-----------+
|See You Again - Wiz Khalifa ft. Charlie Puth|114        |
|Blinding Lights - The Weeknd                  |106        |
|Counting Stars - OneRepublic                  |98         |
|Uptown Funk - Mark Ronson ft. Bruno Mars      |94         |
|Believer - Imagine Dragons                    |87         |
|Bad Guy - Billie Eilish                       |84         |
|Someone Like You - Adele                      |82         |
|Levitating - Dua Lipa                         |77         |
|Shape of You - Ed Sheeran                     |71         |
|Love Story - Taylor Swift                     |63         |
+----------------------------------------------+-----------+

25/11/02 09:38:24 WARN ProcessingTimeExecutor: Current batch is
```

web will get data from API then show us real time results leading songs