Phan Anh Tuan

10/2025

CUSTOMER360

DATA ENGINEER PROJECT REPORT

tuandte27@gmail.com

0943377126

https://github.com/tuandte27

Introduction

Background

Customer 360 View is a comprehensive data record that includes demographic details (name, gender, address, etc.), contact information, preferences, purchase history, and interactions across all touchpoints such as websites, social media, and apps.

It helps businesses understand their customers better, offer more personalized experiences, and improve customer satisfaction.

Data Source

Dataset Information

The dataset of project consists of two main parts: **Interaction Data**, which captures users' interactions with TV programs, and **Behavior Data**, which records users' video search activities. Together, these datasets provide valuable insights into user engagement and viewing preferences for the Customer 360 project.

The Interaction Dataset includes 30 JSON files, each containing one day of user interactions in a month. The Behavior Dataset has 28 Parquet files covering users' video search activity over two months. This structure allows analysis of both daily engagement and longer-term search patterns.

Data Flow

Tools

This project uses tools such as **Apache Spark**, **SimCSE**, **MySQL**, and **Power BI** to handle data processing, storage, and visualization.

- Apache Spark: Used for fast and scalable data processing and transformation.
- SimCSE: Used for text embedding and semantic similarity computations.
- MySQL: Serves as the data warehouse to store and manage structured data.
- Power BI: Used for data visualization and reporting, helping to create dashboards and insights from the data.

ETL Flow

Extract

- Read 30 JSON files and 28 Parquet files, combine them into two DataFrames: interaction data and behavior data.

Transform

For Interaction Data:

- Mapping the AppName field to defined content types.
- Reshaping the dataset and calculate Total Duration by pivoting the data based on Contract and Content Type.
- Calculating key user engagement metrics: Most watch, Taste, Active days, Level Activeness (High, Medium, Low)

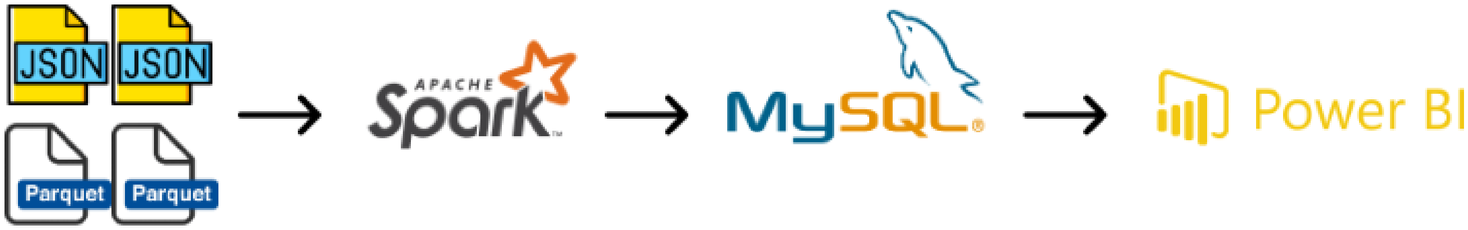
For Behavior Data:

- Identify the most frequently searched keyword for each user in the month.
- Use AI to classify frequent keywords into categories.
- Use SimCSE to match monthly keywords with pre-defined categories based on semantic similarity (over 90%).
- Compare two months of data for each user to find trends in category changes.

Load

- Appending the fully transformed and processed data to the interaction_data table within a MySQL database.

The diagram below depicts the data flow, from ingestion and processing with Spark, storage in MySQL, to visualization in Power BI, as structured by the model.



Output

Schema Interaction

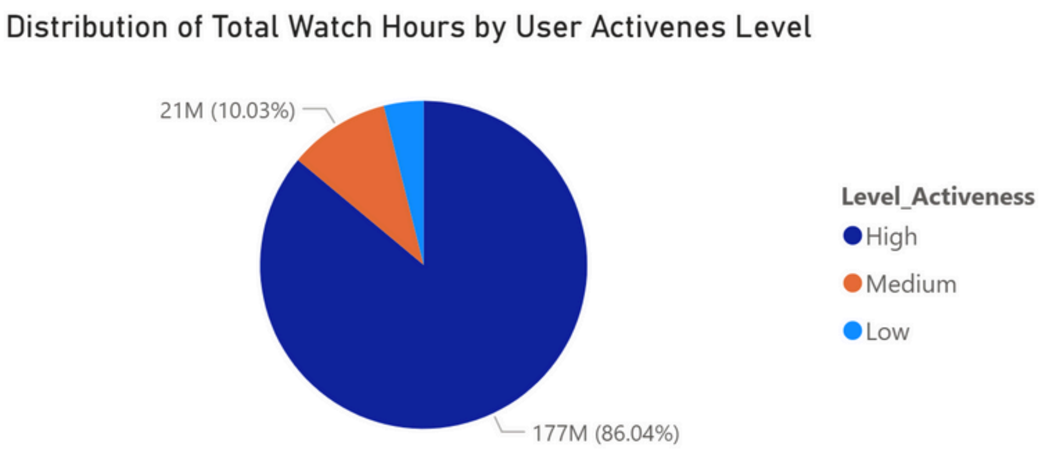
Column	Type	Description
Contract	string	Unique user identifier
Total_Giai_Tri	bigint	Total duration of Entertainment content
Total_Phim_Truyen	bigint	Total duration of Movie content
Total_Truyen_Hinh	bigint	Total duration of Sports content
Total_Thieu_Nhi	bigint	Total duration of Children content
Total_The_Thao	bigint	Total duration of TV content
Most_watched	string	Most-watched content type
Taste	string	Genres the user interacted with
Active	bigint	Number of days the user was active
Level_Activeness	string	User engagement level (High / Medium / Low)

Schema Behavior

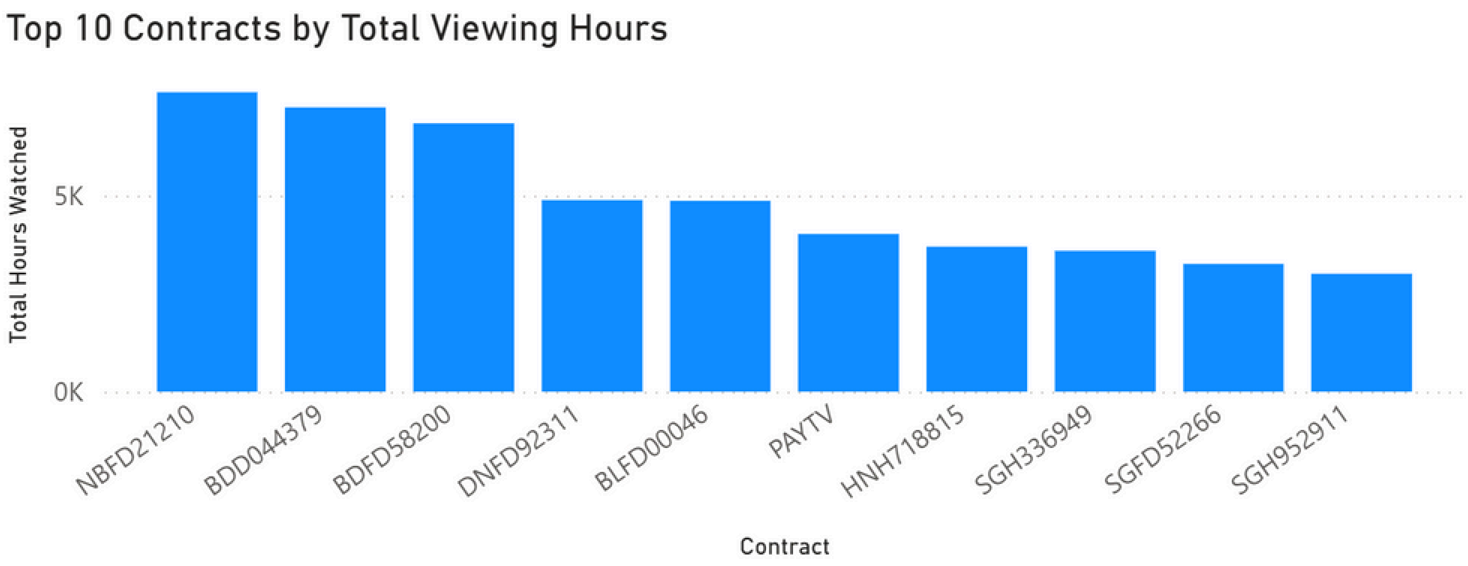
Column	Type	Description
user_id	string	Unique user identifier
most_search_T6	string	User's most frequent search term in June
category_T6	string	User's dominant content category in June
most_search_T7	string	User's most frequent search term in July
category_T7	string	User's dominant content category in July
Trending_Type	string	Indicates if the dominant category Changed or was Unchanged between June and July
Category_Change	string	Details the transition of the dominant category from June to July

Visualizations

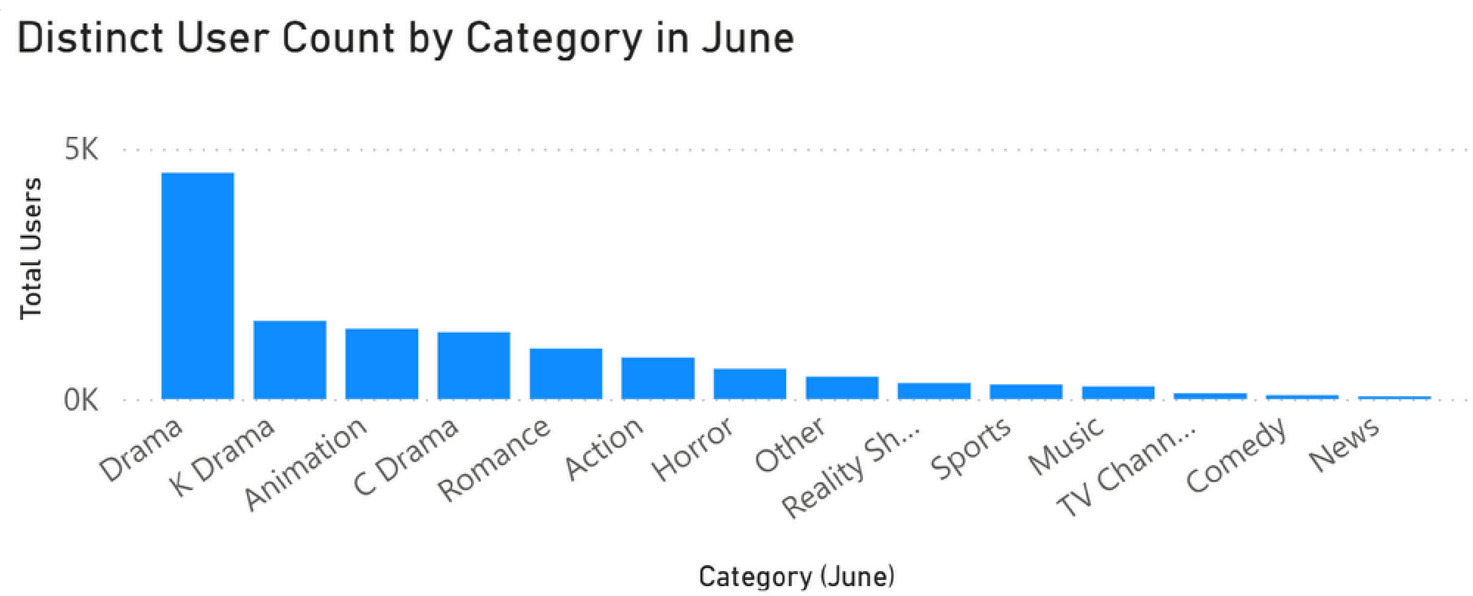
Charts



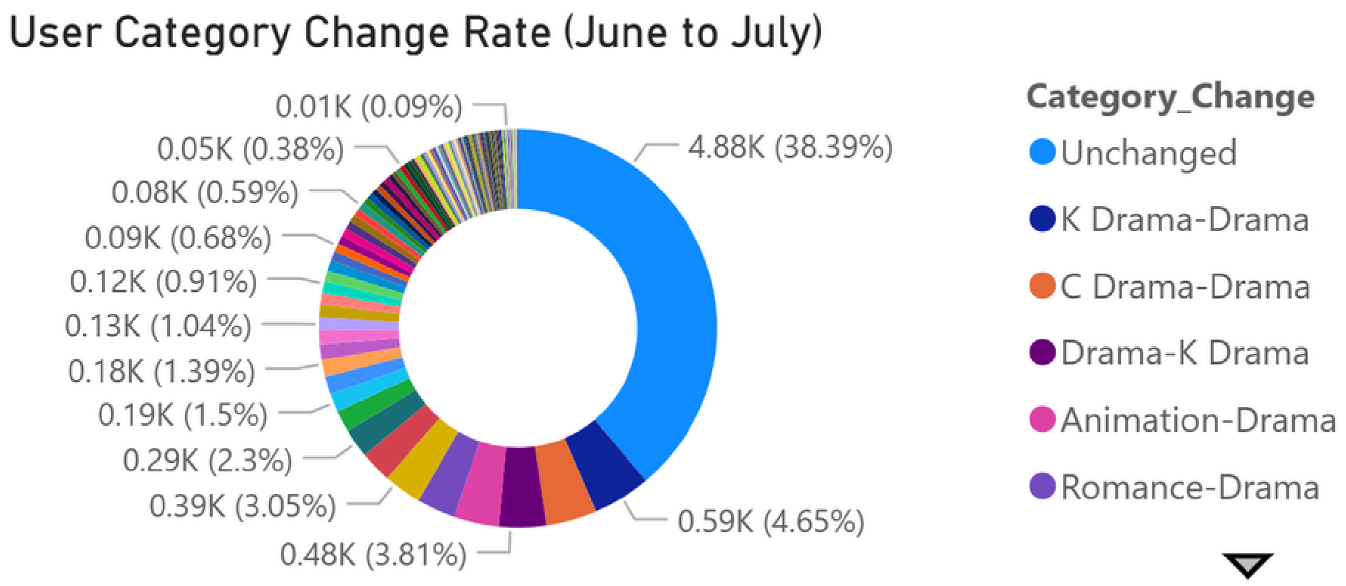
This chart quantifies the platform's total watch hours by user activeness. The High Activeness segment is highly disproportionate, demonstrating the need for the business to focus retention strategies exclusively on this core, high-value user group.



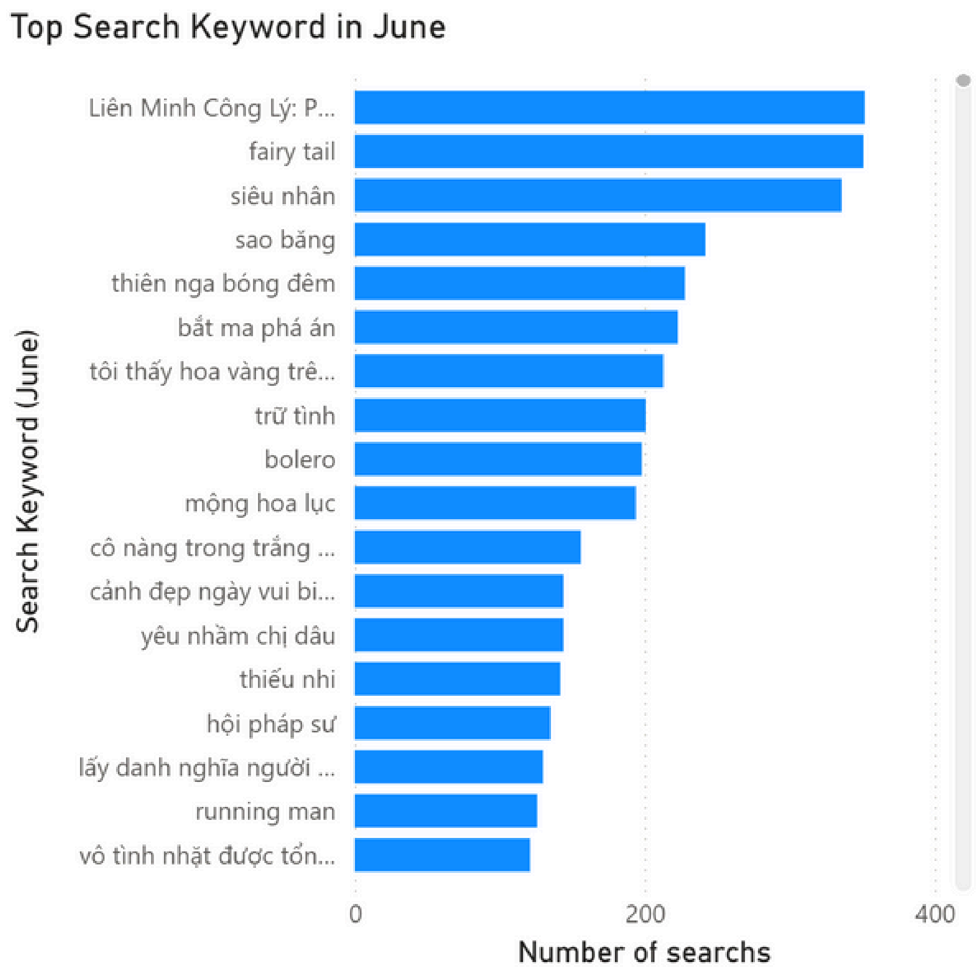
This bar chart highlights the platform's highest-value users based on total viewing hours. The data shows a significant concentration of watch time among a small core group, enabling the business to focus VIP retention efforts and customized service on these high-impact individuals.



This chart identifies content popularity by showing the Distinct User Count per category in June. With 'Drama' dominating the user base, the data validates current content investment and dictates priority for future acquisition and marketing resources.



This donut chart shows user category changes from June to July. Most users (38.39%) stayed in their main category ("Unchanged"), showing strong loyalty to their content preferences. Among those who switched, most moved to the "Drama" category (e.g., K Drama, C Drama), highlighting it as a major destination and key engagement driver.



This chart identifies the top search keywords driving user interest in June, confirming immediate content demands. The high ranking of specific terms (e.g., 'Liên Minh Công Lý...') enables the business to validate its content strategy and prioritize promotional efforts.

Dashboard

Interaction Data

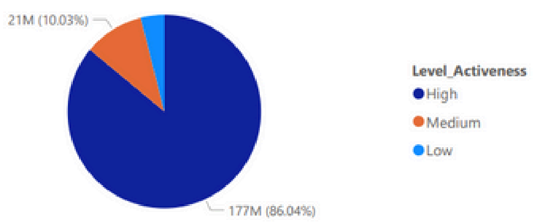
1.92M
Total Customer

205M
Total Hours Watched

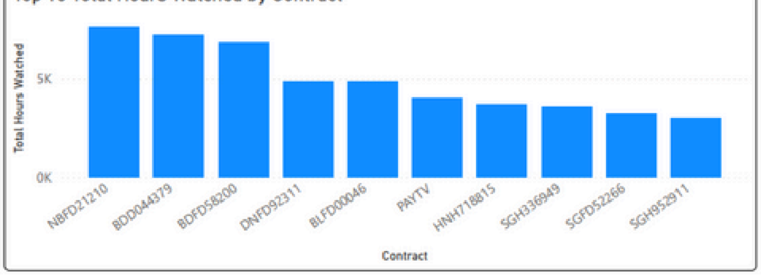
3.56
Average Daily Hours

Truyen Hinh
Most Popular Category

Total Hours Watched by Level_Activeness



Top 10 Total Hours Watched by Contract



Contract	Hours_Giai_Tri	Hours_Phim_Truyen	Hours_The_Thao	Hours_Thieu_Nhi	Hours_Truyen_Hinh	Most Watched	Taste	Level_Activeness
	0.00	0.00	0.00	0.00	2.27	Truyen Hinh	Truyen Hinh	Low
113.183.15.246	0.00	0.00	0.00	0.00	2.28	Truyen Hinh	Truyen Hinh	Low
113.183.60.85	0.00	0.00	0.00	0.00	0.33	Truyen Hinh	Truyen Hinh	Low
113.191.153.52	0.00	0.00	0.00	0.00	2.64	Truyen Hinh	Truyen Hinh	Low
116.107.44.221	0.02	0.02	0.00	0.00	0.05	Truyen Hinh	Giai Tri-Phim Truyen-Truyen Hinh	Low
123.17.50.134	0.00	0.00	0.00	0.00	0.08	Truyen Hinh	Truyen Hinh	Low
123.25.79.212	0.00	0.00	0.00	0.00	3.69	Truyen Hinh	Truyen Hinh	Low
14.166.200.230	0.00	0.00	0.00	0.00	0.00	Truyen Hinh	Truyen Hinh	Low
14.181.179.168	0.00	0.00	0.00	0.00	0.87	Truyen Hinh	Truyen Hinh	Low
14.182.110.125	0.03	0.00	0.00	0.00	0.11	Truyen Hinh	Giai Tri-Truyen Hinh	Low
Total	**158,200.44**	**3,723,134.00**	**31,032.26**	**1,489,621.51**	**199,765,769.55**			

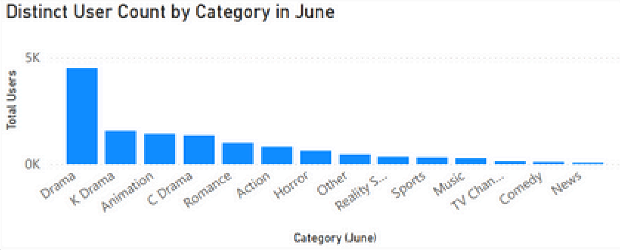
Behavior Data

12.70K
Total Users

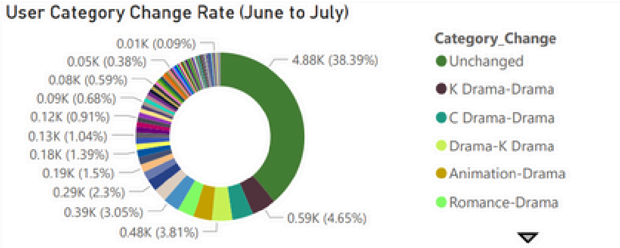
61.61%
Trend Change Rate

Action
Most Popular Category

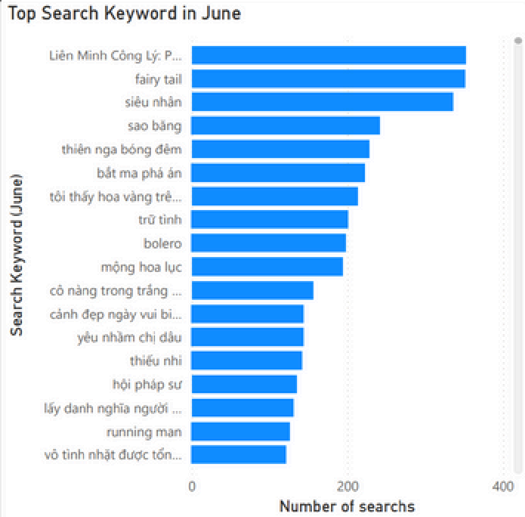
Distinct User Count by Category in June



User Category Change Rate (June to July)



Top Search Keyword in June



user_id	most_search_T6	category_T6	most_search_T7	category_T7	Trending_Type	Category_Change
0002384	ngoc lau xuan	Romance	vẻ đẹp đích thực	K Drama	Changed	Romance-K Drama
0002816	định mệnh anh yêu em	Romance	vân tịch truyện	Drama	Changed	Romance-Drama
0004908	doraemon	Animation	conan	Animation	Unchanged	Unchanged
0014111	running man	Reality Show	cẩm y chi hạ	Sports	Changed	Reality Show-Sports
0015590	tấm cám: chuyện chưa kể	Drama	tấm cám: chuyện chưa kể	Drama	Unchanged	Unchanged
0018591	bolero	Drama	taxi, em tên gì?	Comedy	Changed	Drama-Comedy
0019920	thiếu nhi	Other	bolero	Drama	Changed	Other-Drama
0030912	trái tim bé bỏng	Romance	khủng long	Animation	Changed	Romance-Animation
0031507	cô nàng trong trắng oh woo ri	K Drama	nhất kiến khuynh tâm	Drama	Changed	K Drama-Drama

ETL Code For Interaction Data

```python
from pyspark.sql import SparkSession
from pyspark.sql.functions import *
from pyspark.sql.types import *
import pandas as pd

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

#Phân loại AppName thành các Category
def transform_category(df):
    df = df.withColumn("Type",
             when((col("AppName") == 'CHANNEL') | (col("AppName") =='DSHD')| (col("AppName") =='KPLUS')| (col("AppName") =='KPlus'), "Truyen Hinh")
            .when((col("AppName") == 'VOD') | (col("AppName") =='FIMS_RES')| (col("AppName") =='BHD_RES')|
                   (col("AppName") =='VOD_RES')| (col("AppName") =='FIMS')| (col("AppName") =='BHD')| (col("AppName") =='DANET'), "Phim Truyen")
            .when((col("AppName") == 'RELAX'), "Giai Tri")
            .when((col("AppName") == 'CHILD'), "Thieu Nhi")
            .when((col("AppName") == 'SPORT'), "The Thao")
            .otherwise("Error"))
    df = df.select('Contract','Type','TotalDuration')
    df = df.filter(df.Contract != '0' )
    df = df.filter(df.Type != 'Error')
    return df

def most_watched(df):
    type_cols = ['Total_Truyen_Hinh', 'Total_Phim_Truyen', 'Total_Giai_Tri', 'Total_Thieu_Nhi', 'Total_The_Thao']
    df = df.withColumn('max', greatest(*type_cols)).withColumn('Most Watched',
             when(col('max') == col('Total_Truyen_Hinh'), 'Truyen Hinh')
            .when(col('max') == col('Total_Phim_Truyen'), 'Phim Truyen')
            .when(col('max') == col('Total_Giai_Tri'), 'Giai Tri')
            .when(col('max') == col('Total_Thieu_Nhi'), 'Thieu Nhi')
            .when(col('max') == col('Total_The_Thao'), 'The Thao')
            .otherwise('Error')).drop('max')
    return df

def customer_taste(df):
    df = df.withColumn("Taste",concat_ws("-",
                                     when(col("Total_Giai_Tri") != 0,lit("Giai Tri"))
                                    ,when(col("Total_Phim_Truyen")!= 0,lit("Phim Truyen"))
                                    ,when(col("Total_The_Thao")!= 0,lit("The Thao"))
                                    ,when(col("Total_Thieu_Nhi")!= 0,lit("Thieu Nhi"))
                                    ,when(col("Total_Truyen_Hinh")!= 0,lit("Truyen Hinh"))))
    return df

def agg_and_find_active(df):
    df = df.groupBy("Contract").agg(
        sum("Giai Tri").alias("Total_Giai_Tri"),
        sum("Phim Truyen").alias("Total_Phim_Truyen"),
        sum("The Thao").alias("Total_The_Thao"),
        sum("Thieu Nhi").alias("Total_Thieu_Nhi"),
        sum("Truyen Hinh").alias("Total_Truyen_Hinh"),
        countDistinct("Date").alias("Active")
    )
    df = most_watched(df)
    df = customer_taste(df)
    df = df.withColumn("Level_Activeness",
                       when(col("Active") > 20, "High")\
                       .when((col("Active") <= 20) & (col("Active") >= 10), "Medium")\
                       .otherwise("Low"))

    return df


def etl_1_day(path, path_day):
    print('-------------Reading data from path-------------')
    df = spark.read.json(path + path_day + ".json")
    print('-------------Transforming Category -------------')
    df = df.select("_source.*")
    df = transform_category(df)
    print('-------------Pivoting Data -------------')
    df = df.groupBy("Contract").pivot("Type").sum("TotalDuration").fillna(0)
    df = df.withColumn("Date", to_date(lit(path_day), "yyyyMMdd"))
    return df

def import_to_mysql(result):
    url = 'jdbc:mysql://' + 'localhost' + ':' + '3306' + '/' + 'customer360'
    driver = "com.mysql.cj.jdbc.Driver"
    user = 'root'
    password = ''
    result.write.format('jdbc').option('url',url).option('driver',driver).option('dbtable','interaction_data')\
    .option('user',user).option('password',password).mode('overwrite').save()
    return print("Data Import Successfully")

def main(path):
    start_date="20220401"
    end_date="20220430"
    date_list = pd.date_range(start= start_date ,end = end_date).strftime('%Y%m%d').tolist()
    print("ETL data file " + date_list[0] +".json")
    result = etl_1_day(path, date_list[0])

    for date in date_list[1:]:
        print("ETL data file " + date +".json")
        df_day = etl_1_day(path, date)
        result = result.unionByName(df_day)
    result = result.cache()
    result = result.fillna(0)
    #Tổng hợp và tính chỉ số hoạt động
    result = agg_and_find_active(result)
    #Load vào MySQL
    import_to_mysql(result)

path = "data\\log_content\\"
main(path)
```

ETL Code For Behavior Data

```python
from pyspark.sql import SparkSession
from pyspark.sql.functions import *
from pyspark.sql.window import Window
import pandas as pd
import numpy as np
from sentence_transformers import SentenceTransformer, util

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

model = SentenceTransformer('sentence-transformers/paraphrase-multilingual-MiniLM-L12-v2')

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

def generate_date(start_time, end_time):
    date_list = pd.date_range(start=start_time, end=end_time ).strftime('%Y%m%d').to_list()
    return date_list

def trending_type(df):
    df = df.withColumn("Trending_Type",
        when(col("Category_T6") == col("Category_T7"), "Unchanged")\
        .otherwise("Changed"))
    return df

def category_change(df):
    df = df.withColumn("Category_Change",
        when(col("Category_T6") != col("Category_T7"), concat_ws("-", col("Category_T6"), col("Category_T7"))
        ).otherwise("Unchanged")
    )
    return df

def join_with_simcse(df_spark, category_spark, col_df, col_cat, threshold=0.7):
    """
    Join gần đúng giữa hai dataframe Spark dựa trên độ similar (cosine similarity)
    giữa các chuỗi được mã hóa bằng SimCSE.
    """
    # Convert sang pandas để tính embedding
    df = df_spark.select(col_df).distinct().toPandas()
    cat = category_spark.select(col_cat, "category").distinct().toPandas()

    # Encode text
    df_emb = model.encode(df[col_df].tolist(), convert_to_tensor=True)
    cat_emb = model.encode(cat[col_cat].tolist(), convert_to_tensor=True)

    # Tính cosine similarity
    cosine_scores = util.cos_sim(df_emb, cat_emb).cpu().numpy()

    # Tìm match tốt nhất cho mỗi từ khóa
    best_match = []
    for i, kw in enumerate(df[col_df]):
        j = np.argmax(cosine_scores[i])
        score = cosine_scores[i][j]
        if score >= threshold:
            best_match.append((kw, cat.iloc[j][col_cat], cat.iloc[j]["category"], float(score)))

    matched_df = pd.DataFrame(best_match, columns=[col_df, "matched_keyword", "category", "similarity"])

    # Chuyển ngược sang Spark và rename để tránh trùng cột
    matched_spark = spark.createDataFrame(matched_df)
    matched_spark = matched_spark.withColumnRenamed(col_df, f"{col_df}_key") \
                                 .withColumnRenamed("category", f"category_{col_df}")

    result = df_spark.join(
        matched_spark,
        df_spark[col_df] == matched_spark[f"{col_df}_key"],
        how="inner"
    ).drop(f"{col_df}_key")
    return result
```

```python
def import_to_mysql(result):
    url = 'jdbc:mysql://' + 'localhost' + ':' + '3306' + '/' + 'customer360'
    driver = "com.mysql.cj.jdbc.Driver"
    user = 'root'
    password = ''
    result.write.format('jdbc') \
        .option('url', url) \
        .option('driver', driver) \
        .option('dbtable', 'summary_search_data') \
        .option('user', user) \
        .option('password', password) \
        .mode('overwrite') \
        .save()
    print("Data Import Successfully")

def main(path):
    start_date_T6 = "20220601"
    end_date_T6 = "20220614"
    start_date_T7 = "20220701"
    end_date_T7 = "20220714"
    date_list_T6 = generate_date(start_date_T6, end_date_T6)
    date_list_T7 = generate_date(start_date_T7, end_date_T7)

    # Load data T6
    print("------------Read data T6------------")
    df_T6 = spark.read.parquet(path + date_list_T6[0])
    for date in date_list_T6[1:]:
        df_T6 = df_T6.union(spark.read.parquet(path + date))
    df_T6.cache()
    df_T6 = most_search(df_T6)
    df_T6 = df_T6.withColumnRenamed("most_search", "most_search_T6")

    # Load data T7
    print("------------Read data T7------------")
    df_T7 = spark.read.parquet(path + date_list_T7[0])
    for date in date_list_T7[1:]:
        df_T7 = df_T7.union(spark.read.parquet(path + date))
    df_T7.cache()
    df_T7 = most_search(df_T7)
    df_T7 = df_T7.withColumnRenamed("most_search", "most_search_T7")

    # Đọc file category_search.csv
    category = spark.read.csv("data\\category_search.csv", header=True)

    # Join category với dữ liệu 2 tháng, độ chính xác trên 0.9
    df_T6 = join_with_simcse(df_T6, category, "most_search_T6", "most_search", threshold=0.9)
    df_T7 = join_with_simcse(df_T7, category, "most_search_T7", "most_search", threshold=0.9)

    df_T6 = df_T6.withColumnRenamed("category_most_search_T6", "category_T6")
    df_T7 = df_T7.withColumnRenamed("category_most_search_T7", "category_T7")

    # Join dữ liệu 2 tháng
    result = df_T6.join(df_T7, on="user_id", how="inner")
    result = result.select("user_id", "most_search_T6", "category_T6", "most_search_T7", "category_T7")
    result = trending_type(result)
    result = category_change(result)

    # Load vào MySQL
    import_to_mysql(result)

path = "data\\log_search\\"
main(path)

```

Knowledge Gained

- Gained hands-on experience in building a complete ETL pipeline using Apache Spark for large-scale JSON and Parquet data.
- Learned how to integrate MySQL as a data warehouse for storing transformed data.
- Developed skills in data visualization and dashboard creation using Power BI.
- Improved understanding of data flow, transformation logic, and performance optimization in Spark.