

CUSTOMER360

DATA ENGINEER PROJECT REPORT

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**, **AI model**, **MySQL**, and **Power BI** to handle data processing, storage, and visualization.

- Apache Spark: Used for fast and scalable data processing and transformation.
- AI model: Used for automatic data classification and enrichment, enhancing the overall quality and value of the dataset.
- 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 an AI model to classify users' most frequent search terms in each month into content categories.
- 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)

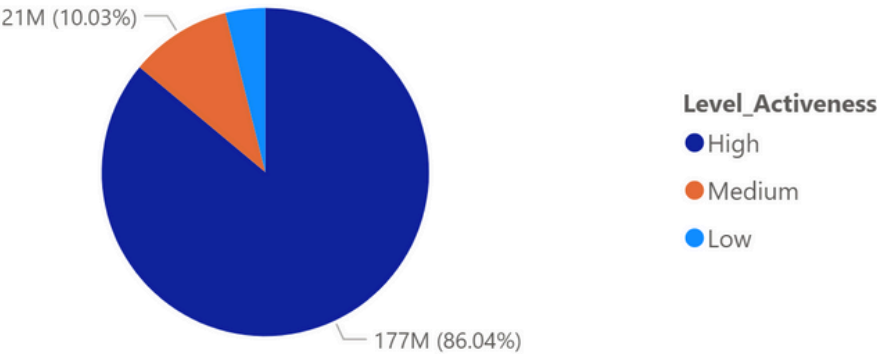
Schema Behavior

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

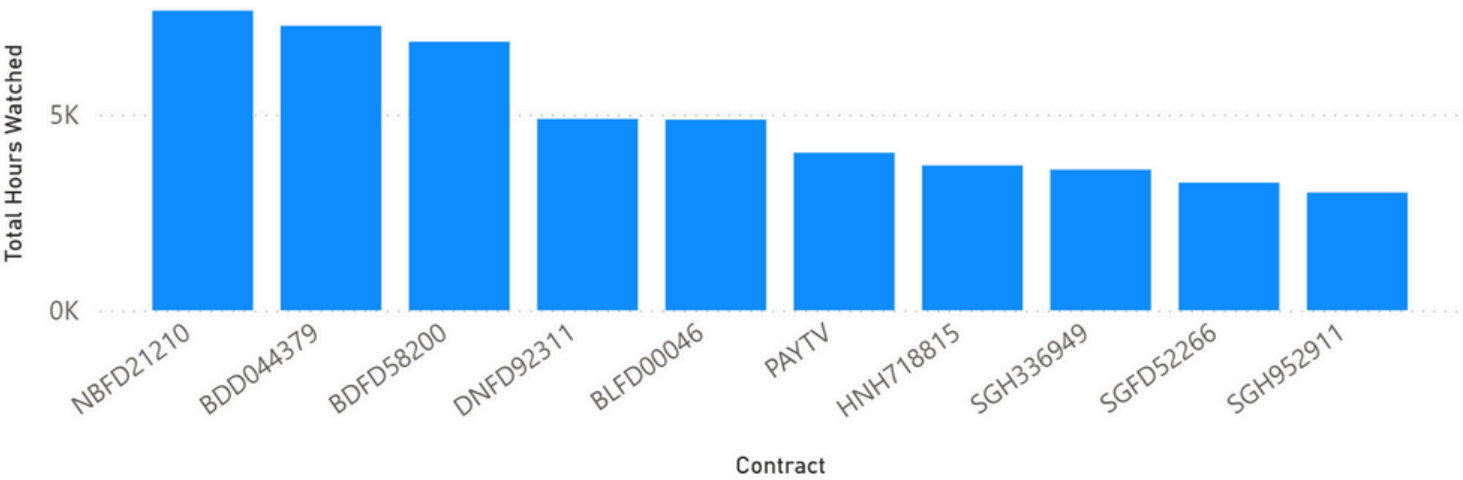
Charts

Distribution of Total Watch Hours by User Activeness Level



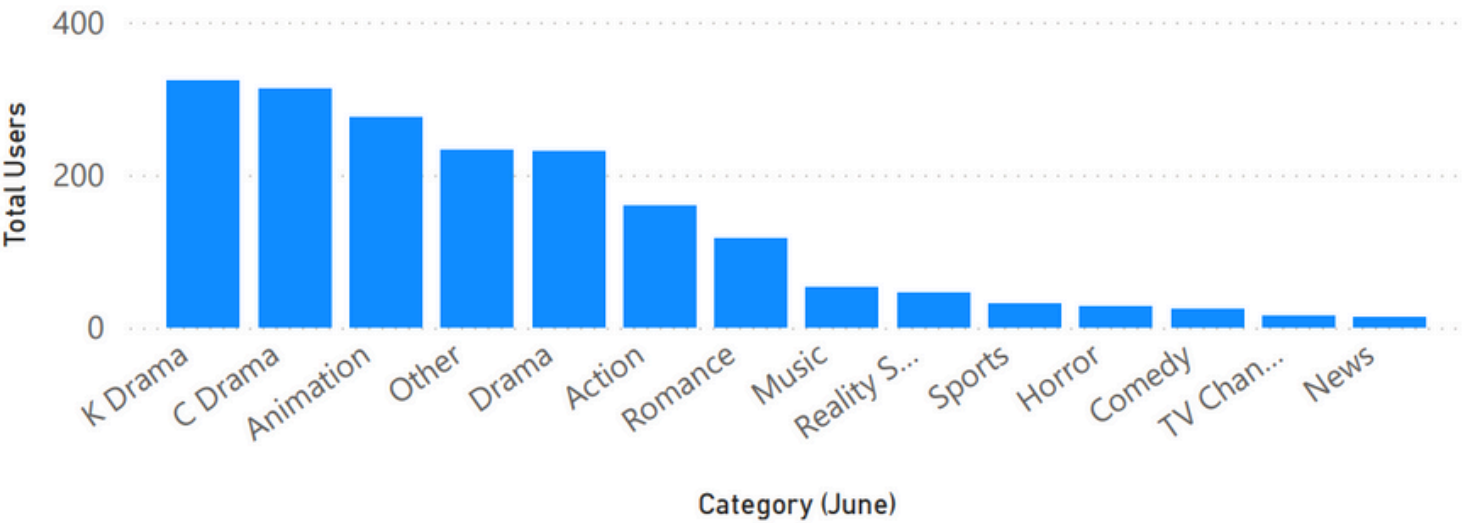
This chart quantifies the platform's total watch hours by user activeness. The High Activeness segment is highly disproportionate, demonstrating the need for the business to focus retention strategies exclusively on this core, high-value user group.

Top 10 Contracts by Total Viewing Hours



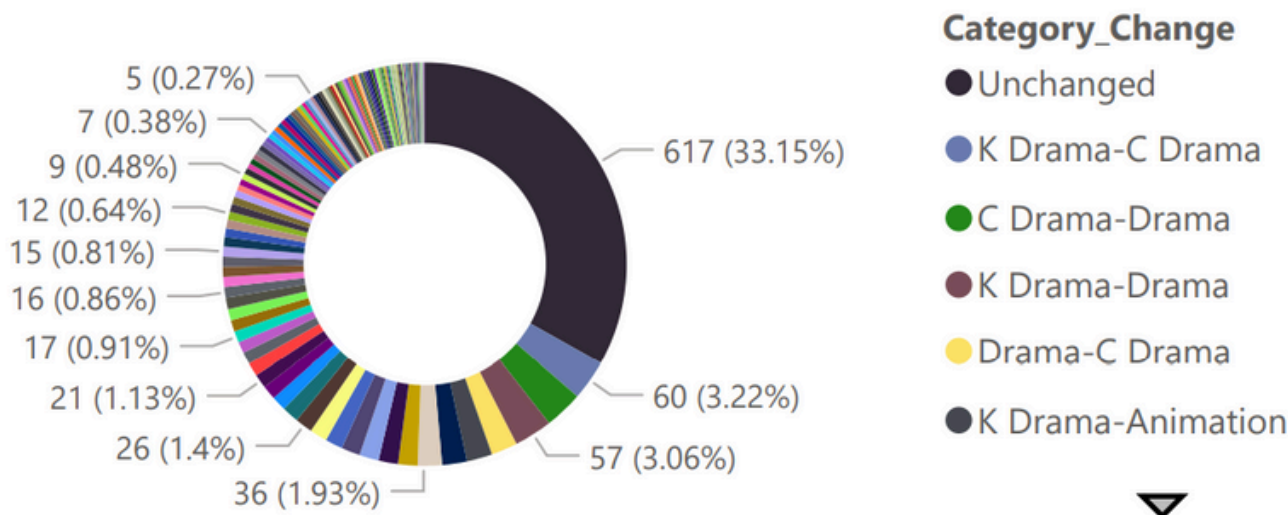
This bar chart highlights the platform's highest-value users based on total viewing hours. The data shows a significant concentration of watch time among a small core group, enabling the business to focus VIP retention efforts and customized service on these high-impact individuals.

Distinct User Count by Category in June



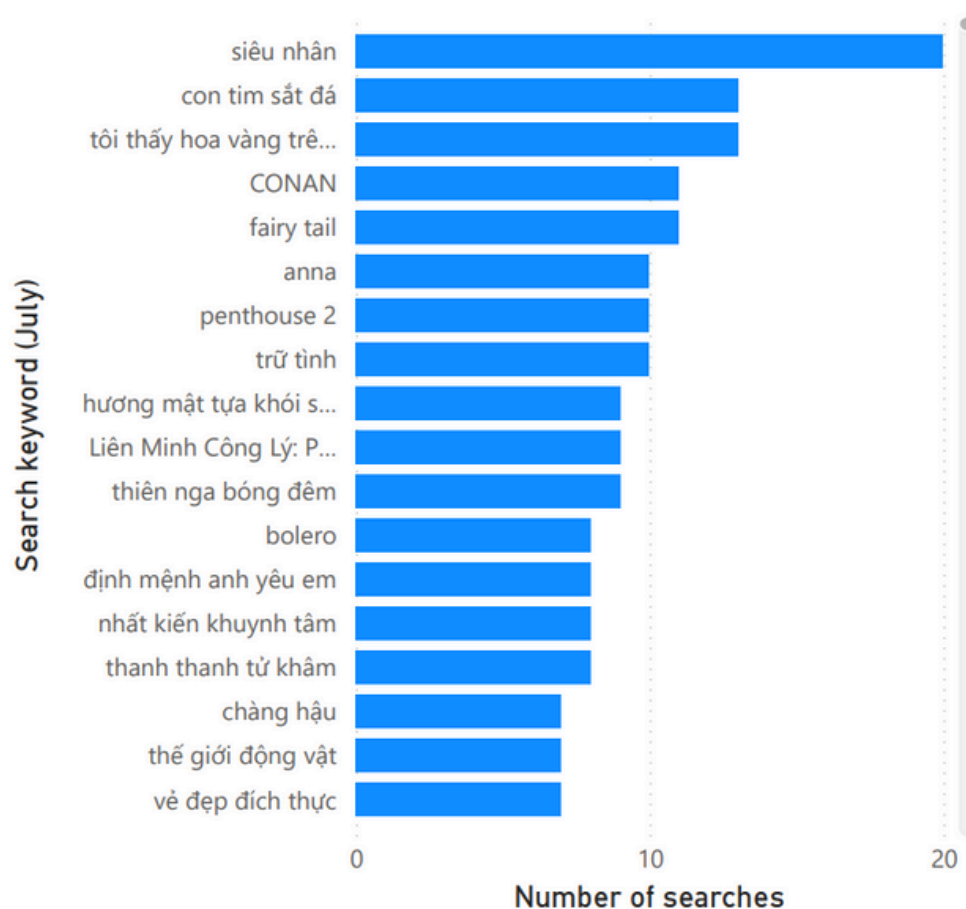
This chart identifies content popularity by showing the Distinct User Count per category in June. With 'K Drama' dominating the user base, the data validates current content investment and dictates priority for future acquisition and marketing resources.

User Category Change Rate (June to July)



This donut chart shows user category changes from June to July. Most users (33.15%) 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.

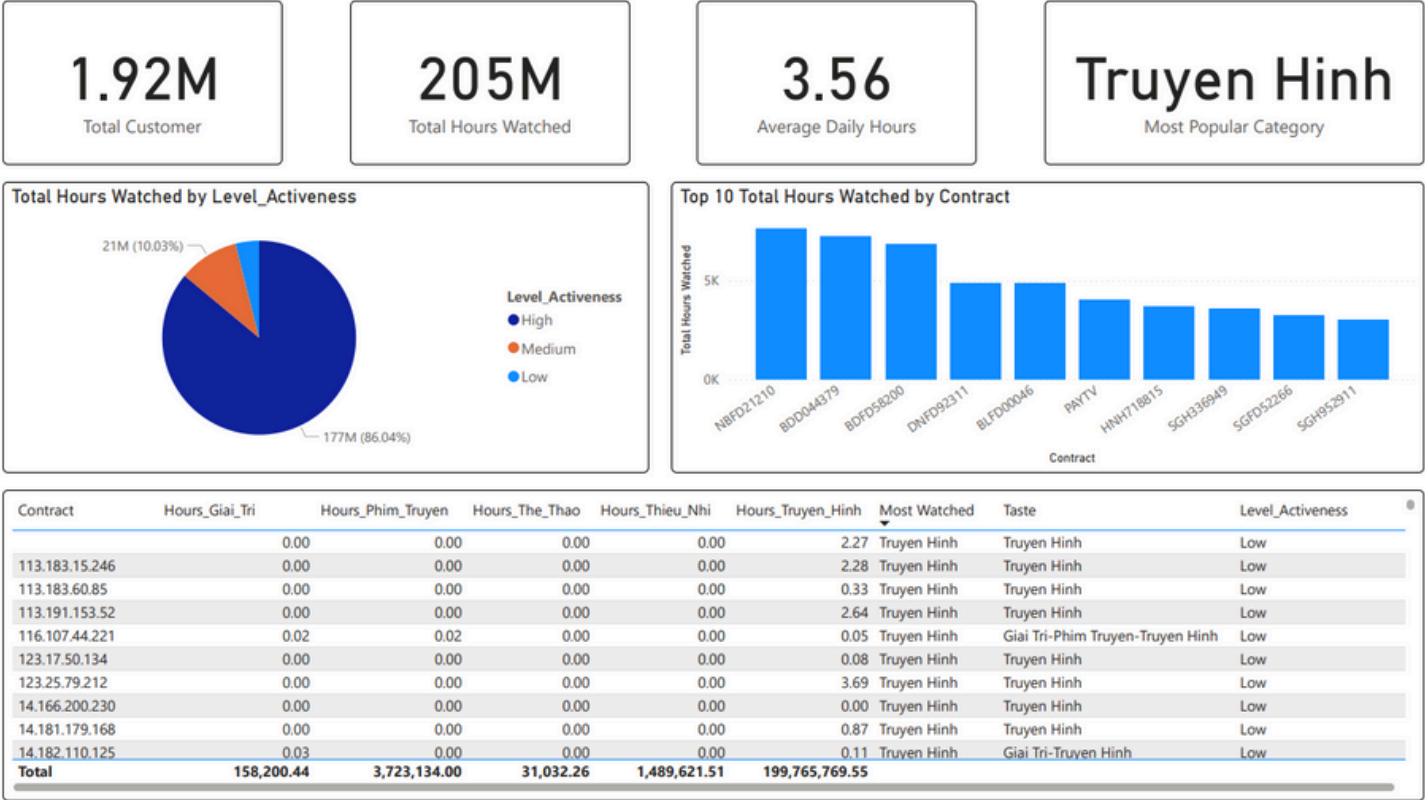
Top Search Keyword in July



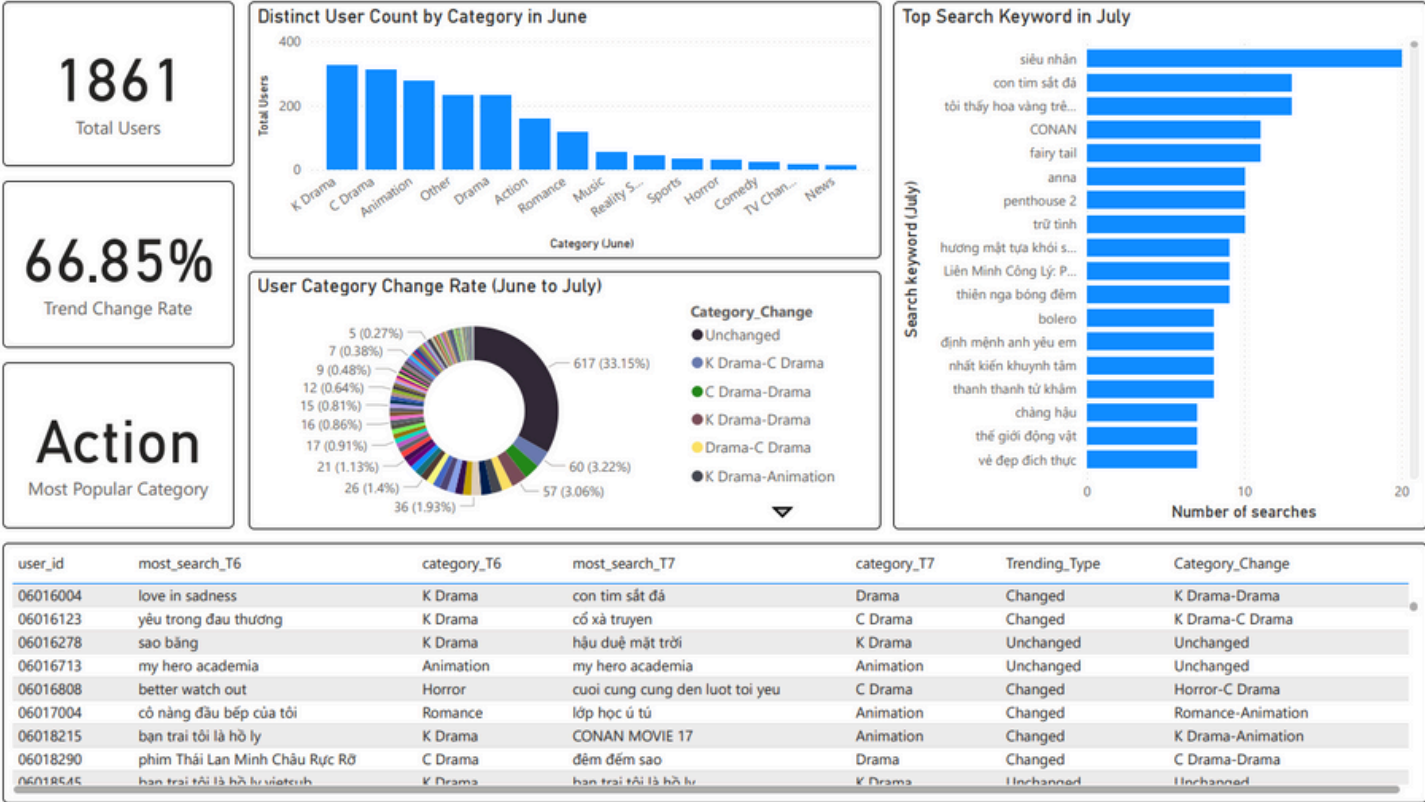
This chart identifies the top search keywords driving user interest in July, confirming immediate content demands. The high ranking of specific terms (e.g., 'siêu nhân') enables the business to validate its content strategy and prioritize promotional efforts.

Dashboard

Interaction Data



Behavior Data



ETL Code For Interaction Data

```
1 from pyspark.sql import SparkSession
2 from pyspark.sql.functions import *
3 from pyspark.sql.types import *
4 import pandas as pd
5
6 spark = SparkSession.builder.config("spark.driver.memory", "8g").getOrCreate()
7
8 #Phân loại AppName thành các Category
9 def transform_category(df):
10     df = df.withColumn("Type",
11         when((col("AppName") == 'CHANNEL') | (col("AppName") == 'DSHD') | (col("AppName") == 'KPLUS') | (col("AppName") == 'KPlus'), "Truyen Hinh")
12         .when((col("AppName") == 'VOD') | (col("AppName") == 'FIMS_RES') | (col("AppName") == 'BHD_RES') |
13             (col("AppName") == 'VOD_RES') | (col("AppName") == 'FIMS') | (col("AppName") == 'BHD') | (col("AppName") == 'DANET'), "Phim Truyen")
14         .when((col("AppName") == 'RELAX'), "Giai Tri")
15         .when((col("AppName") == 'CHILD'), "Thieu Nhi")
16         .when((col("AppName") == 'SPORT'), "The Thao")
17         .otherwise("Error"))
18     df = df.select('Contract', 'Type', 'TotalDuration')
19     df = df.filter(df.Contract != '0')
20     df = df.filter(df.Type != 'Error')
21     return df
22
23 def most_watched(df):
24     type_cols = ['Total_Truyen_Hinh', 'Total_Phim_Truyen', 'Total_Giai_Tri', 'Total_Thieu_Nhi', 'Total_The_Thao']
25     df = df.withColumn('max', greatest(*type_cols)).withColumn('Most Watched',
26         when(col('max') == col('Total_Truyen_Hinh'), 'Truyen Hinh')
27         .when(col('max') == col('Total_Phim_Truyen'), 'Phim Truyen')
28         .when(col('max') == col('Total_Giai_Tri'), 'Giai Tri')
29         .when(col('max') == col('Total_Thieu_Nhi'), 'Thieu Nhi')
30         .when(col('max') == col('Total_The_Thao'), 'The Thao')
31         .otherwise('Error')).drop('max')
32     return df
33
34 def customer_taste(df):
35     df = df.withColumn("Taste", concat_ws("-",
36         when(col("Total_Giai_Tri") != 0, lit("Giai Tri"))
37         ,when(col("Total_Phim_Truyen") != 0, lit("Phim Truyen"))
38         ,when(col("Total_The_Thao") != 0, lit("The Thao"))
39         ,when(col("Total_Thieu_Nhi") != 0, lit("Thieu Nhi"))
40         ,when(col("Total_Truyen_Hinh") != 0, lit("Truyen Hinh"))))
41     return df
42
43 def agg_and_find_active(df):
44     df = df.groupBy("Contract").agg(
45         sum("Giai Tri").alias("Total_Giai_Tri"),
46         sum("Phim Truyen").alias("Total_Phim_Truyen"),
47         sum("The Thao").alias("Total_The_Thao"),
48         sum("Thieu Nhi").alias("Total_Thieu_Nhi"),
49         sum("Truyen Hinh").alias("Total_Truyen_Hinh"),
50         countDistinct("Date").alias("Active")
51     )
52     df = most_watched(df)
53     df = customer_taste(df)
54     df = df.withColumn("Level_Activeness",
55         when(col("Active") > 20, "High")\
56         .when((col("Active") <= 20) & (col("Active") >= 10), "Medium")\
57         .otherwise("Low"))
58     return df
59
60
61
62 def etl_1_day(path, path_day):
63     print('-----Reading data from path-----')
64     df = spark.read.json(path + path_day + ".json")
65     print('-----Transforming Category -----')
66     df = df.select("_source.*")
67     df = transform_category(df)
68     print('-----Pivoting Data -----')
69     df = df.groupBy("Contract").pivot("Type").sum("TotalDuration").fillna(0)
70     df = df.withColumn("Date", to_date(lit(path_day), "yyyyMMdd"))
71     return df
72
73 def import_to_mysql(result):
74     url = 'jdbc:mysql://'+ 'localhost' + ':' + '3306' + '/' + 'customer360'
75     driver = "com.mysql.cj.jdbc.Driver"
76     user = 'root'
77     password = ''
78     result.write.format('jdbc').option('url', url).option('driver', driver).option('dbtable', 'interaction_data')\
79     .option('user', user).option('password', password).mode('overwrite').save()
80     return print("Data Import Successfully")
81
82 def main(path):
83     start_date="20220401"
84     end_date="20220430"
85     date_list = pd.date_range(start= start_date ,end = end_date).strftime('%Y%m%d').tolist()
86     print("ETL data file " + date_list[0] + ".json")
87     result = etl_1_day(path, date_list[0])
88
89     for date in date_list[1:]:
90         print("ETL data file " + date + ".json")
91         df_day = etl_1_day(path, date)
92         result = result.unionByName(df_day)
93     result = result.cache()
94     result = result.fillna(0)
95     #Tổng hợp và tính chỉ số hoạt động
96     result = agg_and_find_active(result)
97     #Load vào MySQL
98     import_to_mysql(result)
99
100 path = "data\\log_content\\"
101 main(path)
```

ETL Code For Behavior Data

```
1 from pyspark.sql import SparkSession
2 from pyspark.sql.functions import *
3 from pyspark.sql.window import Window
4 import pandas as pd
5 import json
6 import google.generativeai as genai
7
8 spark = SparkSession.builder.config("spark.driver.memory", "8g").getOrCreate()
9
10 def most_search(data):
11     data = data.groupBy("user_id", "keyword").count()
12     data = data.orderBy(col("count").desc())
13     data = data.withColumn("rank", row_number().over(Window.partitionBy("user_id").orderBy(col("count").desc()))
14     data = data.filter(col("rank") == 1).select("user_id", "keyword").withColumnRenamed("keyword", "most_search")
15     return data
16
17 def generate_date(start_time, end_time):
18     date_list = pd.date_range(start=start_time, end=end_time).strftime('%Y%m%d').to_list()
19     return date_list
20
21 def trending_type(df):
22     df = df.withColumn("Trending_Type",
23                       when(col("Category_T6") == col("Category_T7"), "Unchanged")\
24                       .otherwise("Changed"))
25     return df
26
27 def generate_category(df, col_name, batch_size=300):
28     GOOGLE_API_KEY = "AIzaSyBe8OW5oMIBniw5s36h_WCA-R8n4EajSWg"
29     genai.configure(api_key=GOOGLE_API_KEY)
30     model = genai.GenerativeModel('gemini-2.5-flash')
31     print("Đã kết nối với Google AI Studio (Gemini 2.5 Flash)")
32
33     df = df.limit(10000).toPandas()
34
35     if col_name not in df.columns:
36         raise ValueError(f"Cột '{col_name}' không tồn tại trong dataframe")
37
38     df.rename(columns={col_name: 'keyword'}, inplace=True)
39
40     unique_keywords = df['keyword'].dropna().astype(str).unique().tolist()
41
42     if not unique_keywords:
43         df['category'] = "Other"
44         return df
45
46     all_mappings = {}
47     total_batches = (len(unique_keywords) + batch_size - 1) // batch_size
48
49     for i in range(0, len(unique_keywords), batch_size):
50         current_batch = unique_keywords[i:i + batch_size]
51         batch_index = i // batch_size + 1
52         print(f"Đang xử lý Batch {batch_index}/{total_batches} ({len(current_batch)} keywords)")
53
54         prompt = f"""
55         Bạn là một chuyên gia phân loại nội dung phim, chương trình truyền hình và các loại nội dung giải trí.
56         Bạn sẽ nhận một danh sách tên có thể viết sai, viết liền không dấu, viết tắt, hoặc chỉ là cụm từ liên quan
57         đến nội dung.
58
59         ⚠️ Nguyên tắc quan trọng:
60         - Không được trả về "Other" nếu có thể đoán được dù chỉ một phần ý nghĩa.
61         - Luôn cố gắng sửa lỗi, nhận diện tên gần đúng hoặc đoán thể loại gần đúng.
62         - Nếu không chắc → chọn thể loại gần nhất (VD: từ mô tả tình cảm → Romance, tên địa danh thể thao → Sports,
63         chương trình giải trí → Reality Show, v.v.)
64
65         Nhiệm vụ của bạn:
66         1. **Chuẩn hoá tên**: thêm dấu tiếng Việt nếu cần, tách từ, chỉnh chính tả.
67         2. **Nhận diện tên hoặc ý nghĩa gốc gần đúng nhất**. Bao gồm:
68         - Tên phim, series, show, chương trình
69         - Quốc gia / đội tuyển (→ "Sports" hoặc "News")
70         - Từ khoá mô tả nội dung
```

```

1      3. **Gán thể loại phù hợp nhất** trong các nhóm sau:
2      - Action
3      - Romance
4      - Comedy
5      - Horror
6      - Animation
7      - Drama
8      - C Drama
9      - K Drama
10     - Sports
11     - Music
12     - Reality Show
13     - TV Channel
14     - News
15     - Other
16
17     Một số quy tắc gợi ý nhanh:
18     - Có từ "VTV", "HTV", "Channel" → TV Channel
19     - Có "running", "master key", "reality" → Reality Show
20     - Quốc gia, CLB bóng đá, sự kiện thể thao → Sports hoặc News
21     - "sex", "romantic", "love" → Romance
22     - "potter", "hogwarts" → Drama / Fantasy
23     - Tên phim Việt/Trung/Hàn → ưu tiên Drama / C Drama / K Drama
24
25     Chỉ trả về **1 JSON object**.
26     Key = tên gốc trong danh sách.
27     Value = thể loại đã phân loại.
28
29     Ví dụ:
30     {{
31     "thuyetminh": "Other",
32     "bigfoot": "Horror",
33     "capdoi": "Romance",
34     "ARGEN": "Sports",
35     "nhật ký": "Drama",
36     "PENT": "C Drama",
37     "running": "Reality Show",
38     "VTV3": "TV Channel"
39     }}
40
41     Danh sách:
42     {current_batch}
43     ""
44
45     try:
46         resp = model.generate_content(prompt)
47         text = resp.text.strip()
48         start, end = text.find("{", text.rfind("}")
49
50         if start == -1 or end == -1:
51             mapping = {m: "Other" for m in current_batch}
52         else:
53             json_text = text[start:end+1]
54             parsed = json.loads(json_text)
55             mapping = {title: parsed.get(title, "Other") for title in current_batch}
56
57         all_mappings.update(mapping)
58
59     except Exception as e:
60         print(f"LỖI API ở Batch {i // batch_size + 1}: {e}")
61         error_mapping = {m: "Other" for m in current_batch}
62         all_mappings.update(error_mapping)
63
64     df['category'] = df['keyword'].map(lambda x: all_mappings.get(x, "Other"))
65
66     return df

```

```

1 def category_change(df):
2     df = df.withColumn("Category_Change",
3         when(col("Category_T6") != col("Category_T7"), concat_ws("-", col("Category_T6"), col("Category_T7"))
4             ).otherwise("Unchanged")
5     )
6     return df
7
8 def import_to_mysql(result):
9     url = 'jdbc:mysql://' + 'localhost' + ':' + '3306' + '/' + 'customer360'
10    driver = "com.mysql.cj.jdbc.Driver"
11    user = 'root'
12    password = ''
13    result.write.format('jdbc') \
14        .option('url', url) \
15        .option('driver', driver) \
16        .option('dbtable', 'behavior_data') \
17        .option('user', user) \
18        .option('password', password) \
19        .mode('overwrite') \
20        .save()
21    print("Data Import Successfully")
22
23 def main(path):
24     start_date_T6 = "20220601"
25     end_date_T6 = "20220614"
26     start_date_T7 = "20220701"
27     end_date_T7 = "20220714"
28     date_list_T6 = generate_date(start_date_T6, end_date_T6)
29     date_list_T7 = generate_date(start_date_T7, end_date_T7)
30
31     # Load data T6
32     print("-----Read data T6-----")
33     df_T6 = spark.read.parquet(path + date_list_T6[0])
34     for date in date_list_T6[1:]:
35         df_T6 = df_T6.union(spark.read.parquet(path + date))
36     df_T6.cache()
37     df_T6 = most_search(df_T6)
38     df_T6 = df_T6.withColumnRenamed("most_search", "most_search_T6")
39
40     # Load data T7
41     print("-----Read data T7-----")
42     df_T7 = spark.read.parquet(path + date_list_T7[0])
43     for date in date_list_T7[1:]:
44         df_T7 = df_T7.union(spark.read.parquet(path + date))
45     df_T7.cache()
46     df_T7 = most_search(df_T7)
47     df_T7 = df_T7.withColumnRenamed("most_search", "most_search_T7")
48
49     # Sinh category cho 2 tháng
50     df_T6 = generate_category(df_T6, col_name="most_search_T6")
51     df_T7 = generate_category(df_T7, col_name="most_search_T7")
52
53     df_T6 = spark.createDataFrame(df_T6).withColumnRenamed("keyword", "most_search_T6")\
54         .withColumnRenamed("category", "category_T6")
55     df_T7 = spark.createDataFrame(df_T7).withColumnRenamed("keyword", "most_search_T7")\
56         .withColumnRenamed("category", "category_T7")
57
58     # Join dữ liệu 2 tháng
59     result = df_T6.join(df_T7, on="user_id", how="inner")
60     result = result.select("user_id", "most_search_T6", "category_T6", "most_search_T7", "category_T7")
61     print("-----Trending Type-----")
62     result = trending_type(result)
63     print("-----Category Change-----")
64     result = category_change(result)
65     result.show()
66     # Load vào MySQL
67     import_to_mysql(result)
68
69 path = "data\\log_search\\"
70 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.