CUSTOMER BEHAVIOUR AND TRENDING PROJECT

1. Introduction

- We have data on user access duration in April 2022 and user search information from June 1–14, 2022, and July 1–14, 2022, from a telecommunications company providing on-demand online streaming services.

- Requirements:

  + **Customer behaviour:** Based on user access duration, what is the total viewing time for different categories by customers? Which category are the most popular among customers?,...

  + **Customer trending:** Based on user search data, identify which movie categories customers searched for the most during the two time periods. Compare the search habits of customers across these two different periods.

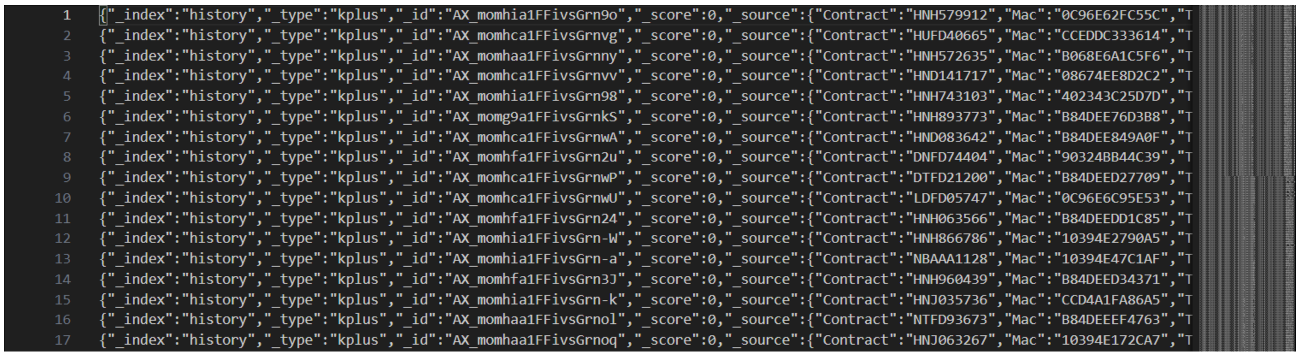
2. Raw Data Overview

2.1. Log_Content Data

a. General Introduction

- The Log_Content data is stored as a JSON source file. It was collected from April 1, 2022, to April 30, 2022, containing information on user history, access duration, and the categories users accessed, with a total of over 50 million data rows.

- The data includes the columns "_index," "_type," "_id," "_score," and "_source." The "_source" column contains information such as: Contract, MAC, TotalDuration, and AppName.



```
{"_index":"history","_type":"kplus","_id":"AX_momhia1FFivsGrn9o","_score":0,"_source":{"Contract":"HNH579912","Mac":"0C96E62FC55C","T
{"_index":"history","_type":"kplus","_id":"AX_momhca1FFivsGrnvg","_score":0,"_source":{"Contract":"HUFD40665","Mac":"CCEDDC333614","T
{"_index":"history","_type":"kplus","_id":"AX_momhaa1FFivsGrnny","_score":0,"_source":{"Contract":"HNH572635","Mac":"B068E6A1C5F6","T
{"_index":"history","_type":"kplus","_id":"AX_momhca1FFivsGrnvv","_score":0,"_source":{"Contract":"HND141717","Mac":"08674EE8D2C2","T
{"_index":"history","_type":"kplus","_id":"AX_momhia1FFivsGrn98","_score":0,"_source":{"Contract":"HNH743103","Mac":"402343C25D7D","T
{"_index":"history","_type":"kplus","_id":"AX_momg9a1FFivsGrnkS","_score":0,"_source":{"Contract":"HNH893773","Mac":"B84DEE76D3B8","T
{"_index":"history","_type":"kplus","_id":"AX_momhca1FFivsGrnwA","_score":0,"_source":{"Contract":"HND083642","Mac":"B84DEE849A0F","T
{"_index":"history","_type":"kplus","_id":"AX_momhfa1FFivsGrn2u","_score":0,"_source":{"Contract":"DNFD74404","Mac":"90324BB44C39","T
{"_index":"history","_type":"kplus","_id":"AX_momhca1FFivsGrnwP","_score":0,"_source":{"Contract":"DTFD21200","Mac":"B84DEED27709","T
{"_index":"history","_type":"kplus","_id":"AX_momhca1FFivsGrnwU","_score":0,"_source":{"Contract":"LDFD05747","Mac":"0C96E6C95E53","T
{"_index":"history","_type":"kplus","_id":"AX_momhfa1FFivsGrn24","_score":0,"_source":{"Contract":"HNH063566","Mac":"B84DEEDD1C85","T
{"_index":"history","_type":"kplus","_id":"AX_momhia1FFivsGrn-W","_score":0,"_source":{"Contract":"HNH866786","Mac":"10394E2790A5","T
{"_index":"history","_type":"kplus","_id":"AX_momhia1FFivsGrn-a","_score":0,"_source":{"Contract":"NBAAA1128","Mac":"10394E47C1AF","T
{"_index":"history","_type":"kplus","_id":"AX_momhfa1FFivsGrn3J","_score":0,"_source":{"Contract":"HNH960439","Mac":"B84DEED34371","T
{"_index":"history","_type":"kplus","_id":"AX_momhia1FFivsGrn-k","_score":0,"_source":{"Contract":"HNJ035736","Mac":"CCD4A1FA86A5","T
{"_index":"history","_type":"kplus","_id":"AX_momhaa1FFivsGrnol","_score":0,"_source":{"Contract":"NTFD93673","Mac":"B84DEEEF4763","T
{"_index":"history","_type":"kplus","_id":"AX_momhaa1FFivsGrnoq","_score":0,"_source":{"Contract":"HNJ063267","Mac":"10394E172CA7","T
```

Picture 01: JSON Data for Log_Content

b. Detailed Overview of the Data Columns

- “Contract” Column: Store information about the reference code.

- “Mac” Column: Store information about the MAC address of the device.

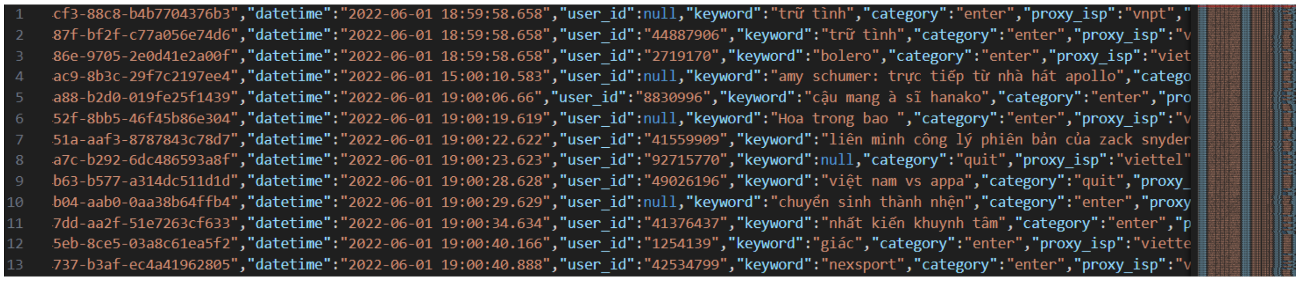
- “TotalDuration” Column: Store information about the user's access duration.

- “AppName” Column: Store information about the name of the application.

2.2. Log_search data

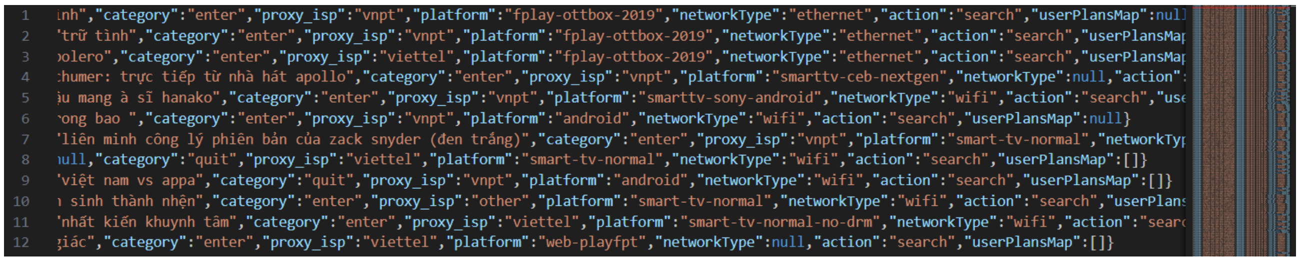
a. General Introduction

- The Log_search data is stored as a PARQUET file, with data collected during two different time periods: (1) From June 1, 2022, to June 14, 2022, and (2) From July 1, 2022, to July 14, 2022, containing information about users' search activities.



```
cf3-88c8-b4b7704376b3","datetime":"2022-06-01 18:59:58.658","user_id":null,"keyword":"trữ tình","category":"enter","proxy_isp":"vnpt","
87f-bf2f-c77a056e74d6","datetime":"2022-06-01 18:59:58.658","user_id":"44887906","keyword":"trữ tình","category":"enter","proxy_isp":"v
86e-9705-2e0d41e2a00f","datetime":"2022-06-01 18:59:58.658","user_id":"2719170","keyword":"bolero","category":"enter","proxy_isp":"viet
ac9-8b3c-29f7c2197ee4","datetime":"2022-06-01 15:00:10.583","user_id":null,"keyword":"amy schumer: trực tiếp từ nhà hát apollo","catego
a88-b2d0-019fe25f1439","datetime":"2022-06-01 19:00:06.66","user_id":"8830996","keyword":"cậu mang à sĩ hanako","category":"enter","pro
52f-8bb5-46f45b86e304","datetime":"2022-06-01 19:00:19.619","user_id":null,"keyword":"Hoa trong bao ","category":"enter","proxy_isp":"v
51a-aaf3-8787843c78d7","datetime":"2022-06-01 19:00:22.622","user_id":"41559909","keyword":"liên minh công lý phiên bản của zack snyder
a7c-b292-6dc486593a8f","datetime":"2022-06-01 19:00:23.623","user_id":"92715770","keyword":null,"category":"quit","proxy_isp":"viettel"
b63-b577-a314dc511d1d","datetime":"2022-06-01 19:00:28.628","user_id":"49026196","keyword":"việt nam vs appa","category":"quit","proxy_
b04-aab0-0aa38b64ffb4","datetime":"2022-06-01 19:00:29.629","user_id":null,"keyword":"chuyển sinh thành nhện","category":"enter","proxy
7dd-aa2f-51e7263cf633","datetime":"2022-06-01 19:00:34.634","user_id":"41376437","keyword":"nhất kiến khuynh tâm","category":"enter","p
5eb-8ce5-03a8c61ea5f2","datetime":"2022-06-01 19:00:40.166","user_id":"1254139","keyword":"giác","category":"enter","proxy_isp":"viette
737-b3af-ec4a41962805","datetime":"2022-06-01 19:00:40.888","user_id":"42534799","keyword":"nexsport","category":"enter","proxy_isp":"v
```

Picture 02: PARQUET Data for Log_search (1)



```
inh","category":"enter","proxy_isp":"vnpt","platform":"fplay-ottbox-2019","networkType":"ethernet","action":"search","userPlansMap":null
'trữ tình","category":"enter","proxy_isp":"vnpt","platform":"fplay-ottbox-2019","networkType":"ethernet","action":"search","userPlansMap
bolero","category":"enter","proxy_isp":"viettel","platform":"fplay-ottbox-2019","networkType":"ethernet","action":"search","userPlansMap
chumer: trực tiếp từ nhà hát apollo","category":"enter","proxy_isp":"vnpt","platform":"smarttv-ceb-nextgen","networkType":null,"action":
ậu mang à sĩ hanako","category":"enter","proxy_isp":"vnpt","platform":"smarttv-sony-android","networkType":"wifi","action":"search","use
rong bao ","category":"enter","proxy_isp":"vnpt","platform":"android","networkType":"wifi","action":"search","userPlansMap":null}
'liên minh công lý phiên bản của zack snyder (đen trắng)","category":"enter","proxy_isp":"vnpt","platform":"smart-tv-normal","networkTyp
null,"category":"quit","proxy_isp":"viettel","platform":"smart-tv-normal","networkType":"wifi","action":"search","userPlansMap":[]}
'việt nam vs appa","category":"quit","proxy_isp":"vnpt","platform":"android","networkType":"wifi","action":"search","userPlansMap":[]}
n sinh thành nhện","category":"enter","proxy_isp":"other","platform":"smart-tv-normal","networkType":"wifi","action":"search","userPlans
'nhất kiến khuynh tâm","category":"enter","proxy_isp":"viettel","platform":"smart-tv-normal-no-drm","networkType":"wifi","action":"searc
giác","category":"enter","proxy_isp":"viettel","platform":"web-playfpt","networkType":null,"action":"search","userPlansMap":[]}
```

Picture 03: PARQUET Data for Log_search (2)

b. Detailed Overview of the Data Columns

Although there are many columns, we will focus on a few important ones used for transformation:

- “user_id” column: Store the ID of the user accessing the system.

- “datetime” column: Store the timestamp of the access.

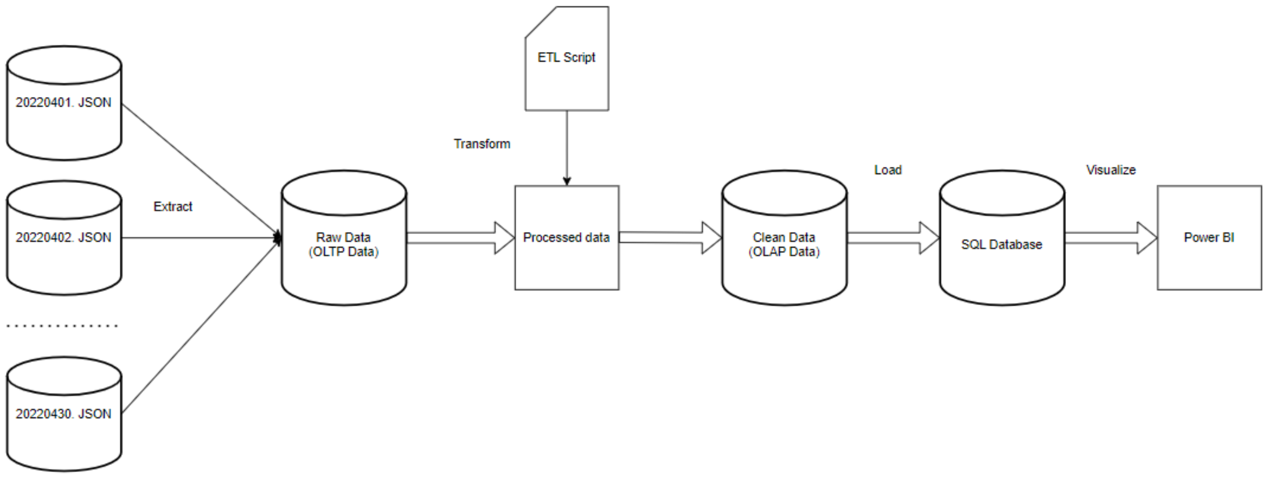
- “keyword” column: Store the search keyword.

- “category” column: Store information about the action, whether the user is "enter" or "quit".

3. Data Pipeline introduction

3.1. Log_content data pipeline

- The purpose of this ETL process is to calculate how much time each user spends on each different category, identify the category that each user spends the most time on, and determine which categories users access. Additionally, it aims to assess customer activeness. From this, we can analyze **"Customer Behavior"**.

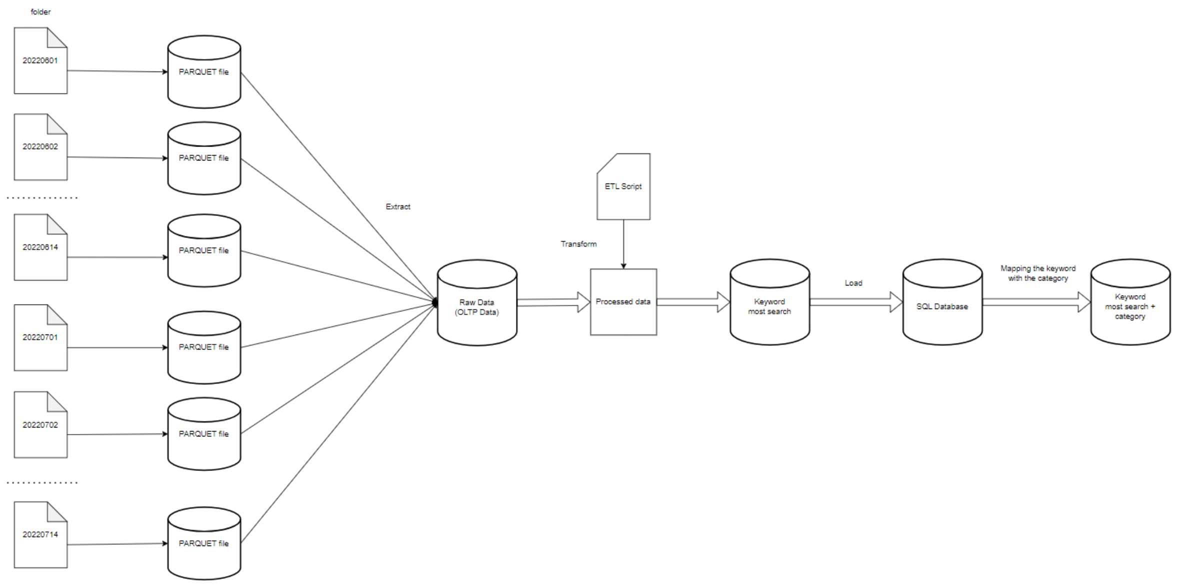


Picture 04: Log_content data pipeline

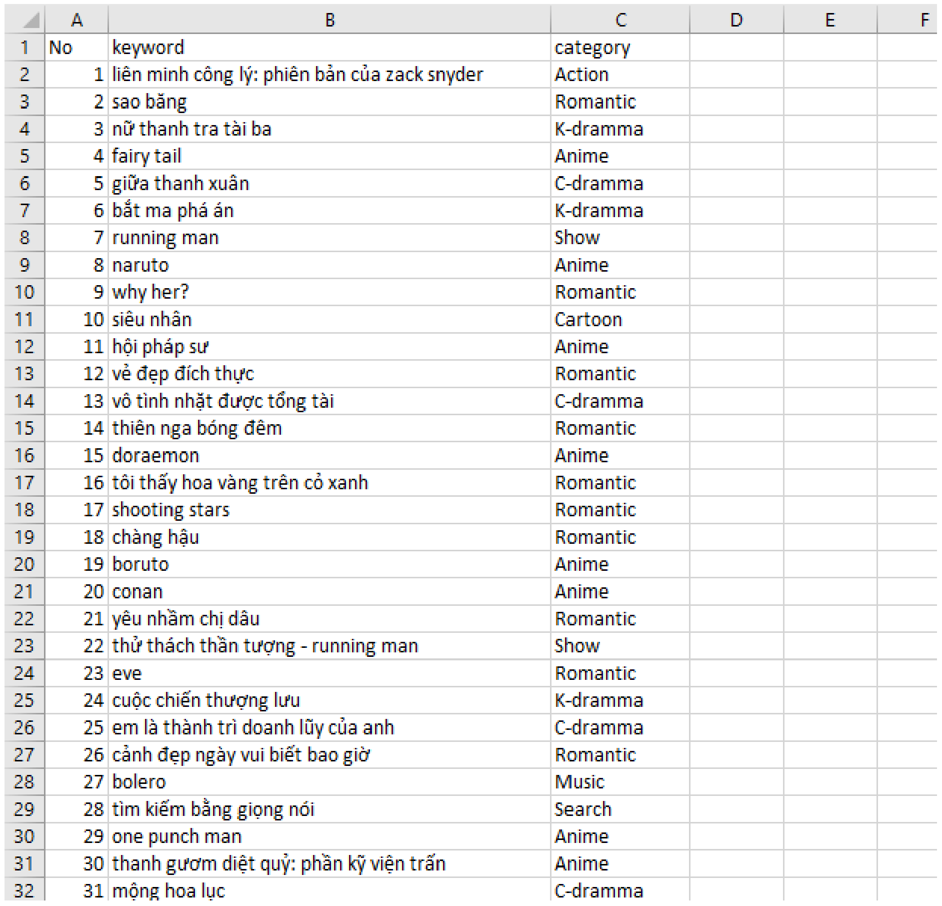
3.2. Log_search data pipeline

3.2.1. Finding keyword most search and mapping with the category

- We identify the keywords that users accessed the most in both time periods, and then select the top 100 most searched keywords. Next, we create a “Category” column, and each keyword will be associated with a specific category. For example, the keyword "fairy tail" will be associated with the "Anime" category. This file will then be used for the main ETL process.



Picture 05: Keyword most search data pipeline

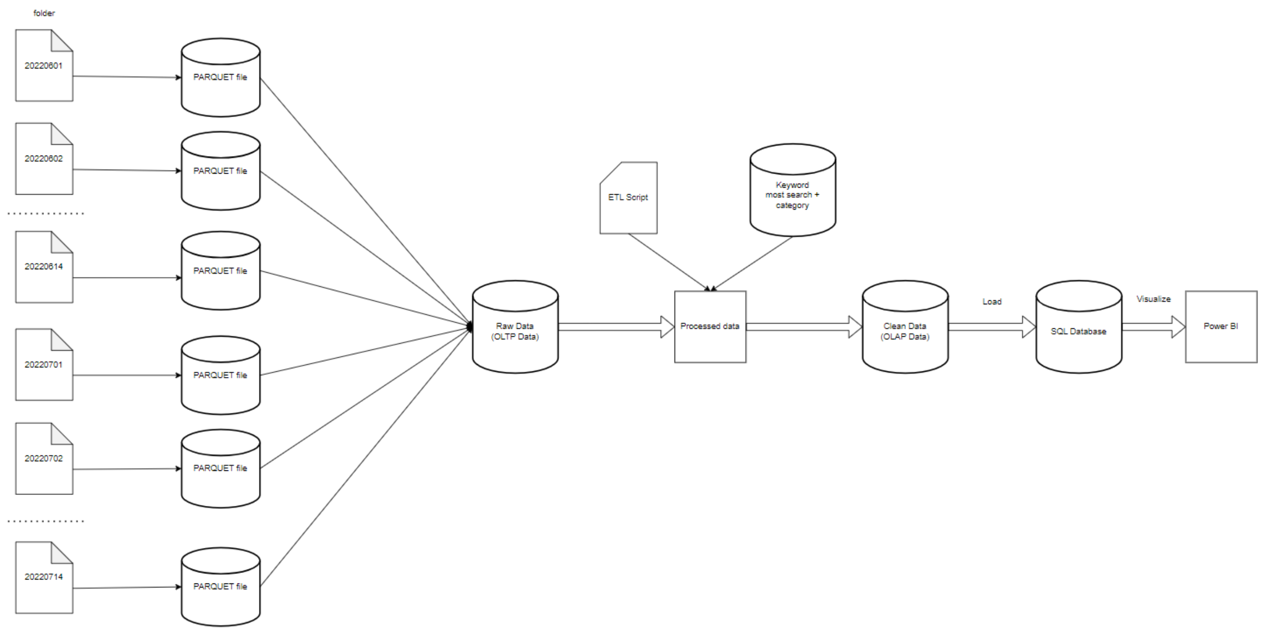


	A	B	C	D	E	F
1	No	keyword	category			
2	1	liên minh công lý: phiên bản của zack snyder	Action			
3	2	sao băng	Romantic			
4	3	nữ thanh tra tài ba	K-dramma			
5	4	fairy tail	Anime			
6	5	giữa thanh xuân	C-dramma			
7	6	bắt ma phá án	K-dramma			
8	7	running man	Show			
9	8	naruto	Anime			
10	9	why her?	Romantic			
11	10	siêu nhân	Cartoon			
12	11	hội pháp sư	Anime			
13	12	vẻ đẹp đích thực	Romantic			
14	13	vô tình nhặt được tổng tài	C-dramma			
15	14	thiên nga bóng đêm	Romantic			
16	15	doraemon	Anime			
17	16	tôi thấy hoa vàng trên cỏ xanh	Romantic			
18	17	shooting stars	Romantic			
19	18	chàng hậu	Romantic			
20	19	boruto	Anime			
21	20	conan	Anime			
22	21	yêu nhầm chị dâu	Romantic			
23	22	thử thách thần tượng - running man	Show			
24	23	eve	Romantic			
25	24	cuộc chiến thượng lưu	K-dramma			
26	25	em là thành trì doanh lũy của anh	C-dramma			
27	26	cảnh đẹp ngày vui biết bao giờ	Romantic			
28	27	bolero	Music			
29	28	tìm kiếm bằng giọng nói	Search			
30	29	one punch man	Anime			
31	30	thanh gươm diệt quỷ: phần kỹ viện trấn	Anime			
32	31	mộng hoa lục	C-dramma			

Picture 06: "Keyword most search + category" file

3.2.2. Log_search data pipeline

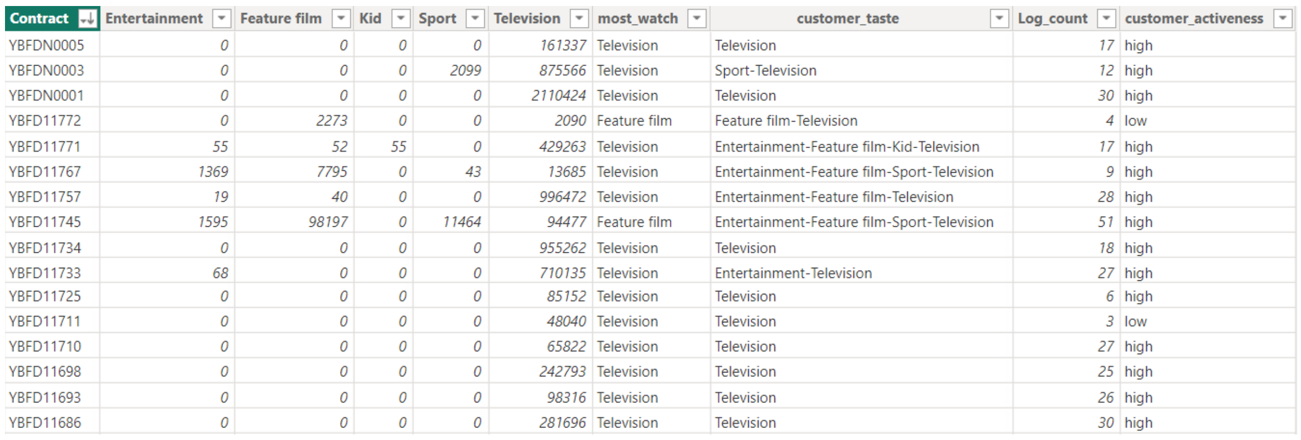
- The purpose of this ETL process is to identify the most searched keywords and their corresponding categories for each user during the two time periods: from June 1–14, 2022, and from July 1–14, 2022. Additionally, it aims to examine whether the categories users are interested in have changed between the two months, and if so, how they have changed. This will enable the analysis of **"Customer Trending"**.



Picture 07: Log_search data pipeline

4. Clean data (OLAP data)

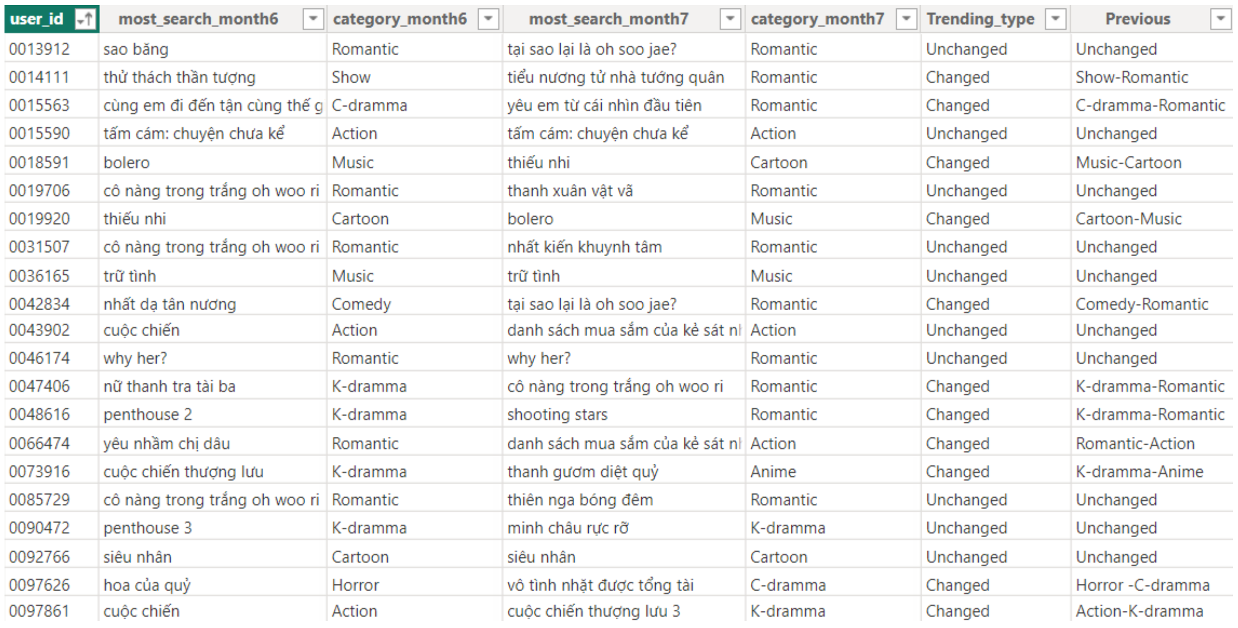
4.1. Log_content clean data



Contract	Entertainment	Feature film	Kid	Sport	Television	most_watch	customer_taste	Log_count	customer_activeness
YBFDN0005	0	0	0	0	161337	Television	Television	17	high
YBFDN0003	0	0	0	2099	875566	Television	Sport-Television	12	high
YBFDN0001	0	0	0	0	2110424	Television	Television	30	high
YBFD11772	0	2273	0	0	2090	Feature film	Feature film-Television	4	low
YBFD11771	55	52	55	0	429263	Television	Entertainment-Feature film-Kid-Television	17	high
YBFD11767	1369	7795	0	43	13685	Television	Entertainment-Feature film-Sport-Television	9	high
YBFD11757	19	40	0	0	996472	Television	Entertainment-Feature film-Television	28	high
YBFD11745	1595	98197	0	11464	94477	Feature film	Entertainment-Feature film-Sport-Television	51	high
YBFD11734	0	0	0	0	955262	Television	Television	18	high
YBFD11733	68	0	0	0	710135	Television	Entertainment-Television	27	high
YBFD11725	0	0	0	0	85152	Television	Television	6	high
YBFD11711	0	0	0	0	48040	Television	Television	3	low
YBFD11710	0	0	0	0	65822	Television	Television	27	high
YBFD11698	0	0	0	0	242793	Television	Television	25	high
YBFD11693	0	0	0	0	98316	Television	Television	26	high
YBFD11686	0	0	0	0	281696	Television	Television	30	high

Picture 08: Log_content clean data

4.2. Log_search clean data

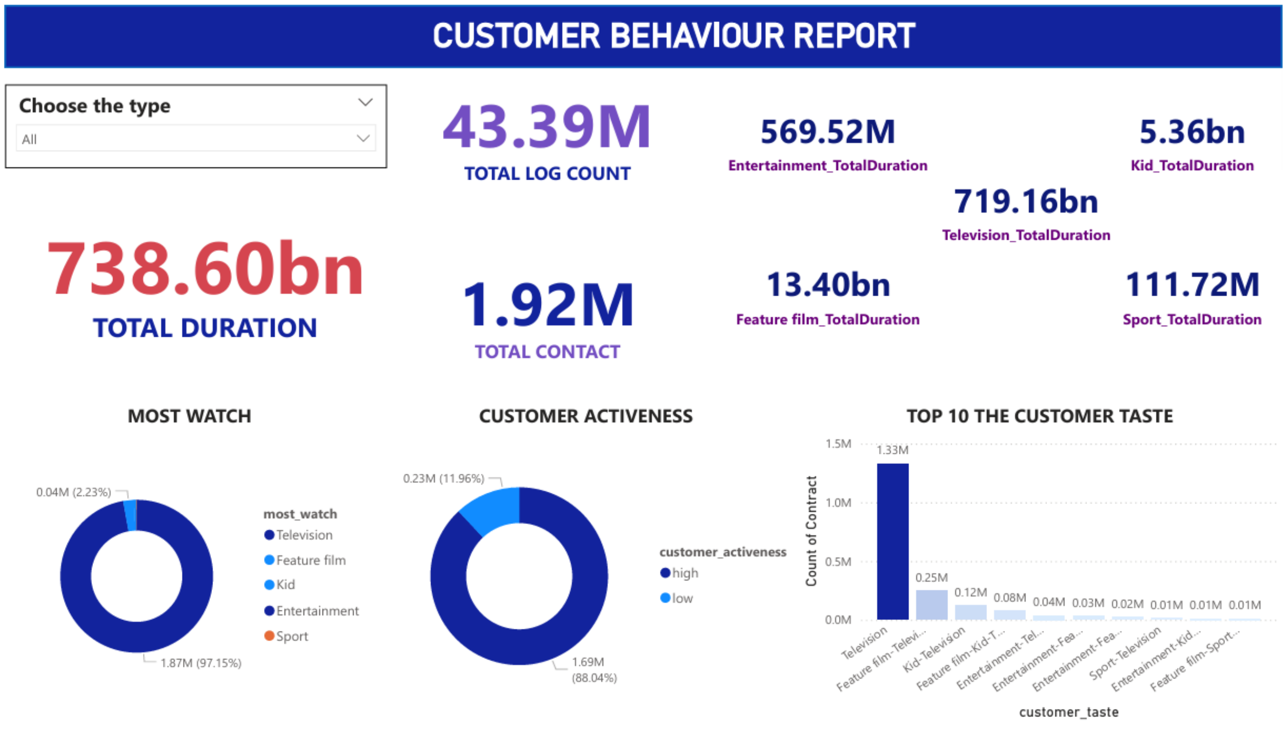


user_id	most_search_month6	category_month6	most_search_month7	category_month7	Trending_type	Previous
0013912	sao băng	Romantic	tại sao lại là oh soo jae?	Romantic	Unchanged	Unchanged
0014111	thử thách thần tượng	Show	tiểu nương tử nhà tướng quân	Romantic	Changed	Show-Romantic
0015563	cùng em đi đến tận cùng thế g	C-dramma	yêu em từ cái nhìn đầu tiên	Romantic	Changed	C-dramma-Romantic
0015590	tấm cám: chuyện chưa kể	Action	tấm cám: chuyện chưa kể	Action	Unchanged	Unchanged
0018591	bolero	Music	thiếu nhi	Cartoon	Changed	Music-Cartoon
0019706	cô nàng trong trắng oh woo ri	Romantic	thanh xuân vật vã	Romantic	Unchanged	Unchanged
0019920	thiếu nhi	Cartoon	bolero	Music	Changed	Cartoon-Music
0031507	cô nàng trong trắng oh woo ri	Romantic	nhất kiến khuynh tâm	Romantic	Unchanged	Unchanged
0036165	trữ tình	Music	trữ tình	Music	Unchanged	Unchanged
0042834	nhất dạ tân nương	Comedy	tại sao lại là oh soo jae?	Romantic	Changed	Comedy-Romantic
0043902	cuộc chiến	Action	danh sách mua sắm của kẻ sát nl	Action	Unchanged	Unchanged
0046174	why her?	Romantic	why her?	Romantic	Unchanged	Unchanged
0047406	nữ thanh tra tài ba	K-dramma	cô nàng trong trắng oh woo ri	Romantic	Changed	K-dramma-Romantic
0048616	penthouse 2	K-dramma	shooting stars	Romantic	Changed	K-dramma-Romantic
0066474	yêu nhầm chị dâu	Romantic	danh sách mua sắm của kẻ sát nl	Action	Changed	Romantic-Action
0073916	cuộc chiến thượng lưu	K-dramma	thanh gươm diệt quỷ	Anime	Changed	K-dramma-Anime
0085729	cô nàng trong trắng oh woo ri	Romantic	thiên nga bóng đêm	Romantic	Unchanged	Unchanged
0090472	penthouse 3	K-dramma	minh châu rực rỡ	K-dramma	Unchanged	Unchanged
0092766	siêu nhân	Cartoon	siêu nhân	Cartoon	Unchanged	Unchanged
0097626	hoa của quỷ	Horror	vô tình nhặt được tổng tài	C-dramma	Changed	Horror -C-dramma
0097861	cuộc chiến	Action	cuộc chiến thượng lưu 3	K-dramma	Changed	Action-K-dramma

Picture 09: Log_search clean data

5. Customer behaviour and trending analyst

5.1. Customer behaviour analyst



Picture 10: Customer behaviour report

- Through the dashboard, there are several key metrics to note:

+ **The total duration** of user access in April 2022 is **738.6 billion seconds**.

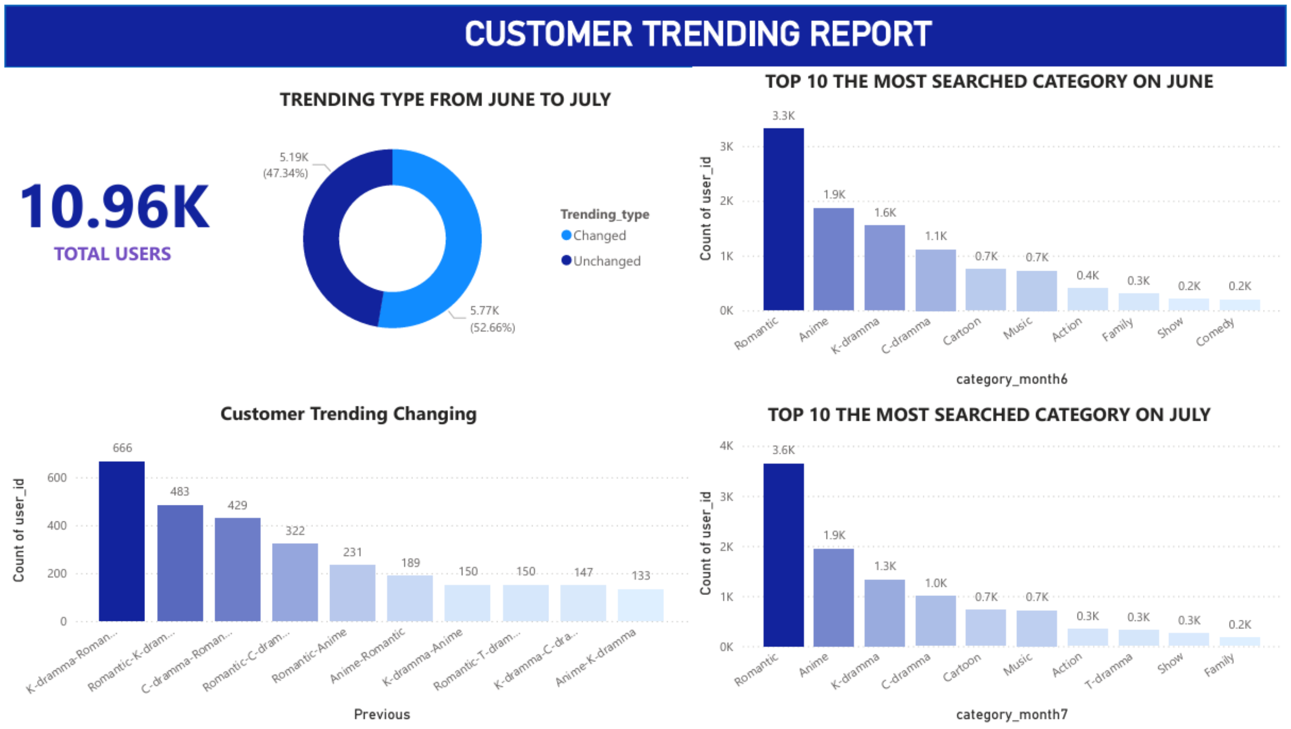
+ **The total number of users** is **1.92 million**, and the total log count is **43.39 million**.

+ **The "Entertainment" category** was accessed for the longest duration, significantly more than other categories, with **719.16 billion seconds** (accounting for **97.15% of the total duration**). In contrast, the "Sport" category had the least access duration, with 111.72 million seconds.

+ **88.04% of users** have **high customer activeness**, which is a positive indicator showing a large number of returning and frequent users.

+ Regarding customer taste, **most users prefer accessing the "Television" category**, with additional interests in other categories such as Feature films and Kids content.

5.2. Customer trending analyst



Picture 11: Customer Trending report

- Through the dashboard, several key metrics to note:

+ **Total users is 10,960**, with **over half (52.66%) changing their preferences**—switching search categories between June and July.

+ The most popular search categories showed little change between the two periods, with **Romantic being the top-searched category**. In June, 3,300 users prioritized searching for it, and in July, there was a slight increase to 3,600 users.

+ Regarding Customer Trending, it is mostly the movement back and forth between users accessing the "Romantic" category and the "Dramma" categories.

6. Code scripts

6.1. Customer behaviour

```
import findspark
findspark.init()
from pyspark.sql import SparkSession
from pyspark.sql.functions import *
from datetime import datetime
from datetime import date, timedelta

spark = SparkSession.builder.getOrCreate()

# Processing the entered time period
def date_transform(start_date,end_date):
    start_date_dt = datetime.strptime(start_date, '%Y%m%d')
    end_date_dt = datetime.strptime(end_date,'%Y%m%d')
    day_list = [(start_date_dt + timedelta(days=i)).strftime('%Y%m%d') for i in 
range((end_date_dt - start_date_dt).days + 1)]
    return(day_list)

# Loading data based on the request time series
def data_extract(day_list,path):
    from pyspark.sql.types import StructType
    df = spark.read.json(path + day_list[0] + '.json')
    df = df.withColumn('Date', lit(datetime.strptime(day_list[0], '%Y%m%d')))
    print('Processed completed {}'.format(day_list[0]))
    print('----------------------')
    for i in day_list[1:]:
        data = spark.read.json(path + i + '.json')
        data = data.withColumn('Date', lit(datetime.strptime(i, '%Y%m%d')))
        df = df.union(data)
        print('Processed completed {}'.format(i))
        print('-----------------------')
    print('Showing data sample')
    print('-----------------------')
    print(df.show(20))
    print('-----------------------')
    print('Showing data structure')
    print('-----------------------')
    df.printSchema()
    print('-----------------------')
    return(df)

# Processing data
def data_transform(df):
    df = df.select("Date", *[col("_source." + c).alias(c) for c in 
df.select("_source.*").columns])
    df = df.withColumn("Type",
        when((col("AppName") == 'CHANNEL') | (col("AppName") =='DSHD')|
(col("AppName") =='KPLUS')|
            (col("AppName") =='KPlus'), "Television")
        .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'), "Feature film")
        .when((col("AppName") == 'RELAX'), "Entertainment")
        .when((col("AppName") == 'CHILD'), "Kid")
        .when((col("AppName") == 'SPORT'), "Sport")
        .otherwise("Error"))
    df = df.select('Contract','Date','Type','TotalDuration')
    df = df.filter(df.Contract != '0')
    df = df.filter(df.Type != 'Error')
    print('Filter out contract = 0 and type is Error')
    print('-----------------------')
    df = df.groupBy('Contract','Type',
'Date').sum('TotalDuration').withColumnRenamed('sum(TotalDuration)','TotalDuratio
n')
    print('Sum TotalDuration according Contract and Type')
    print('-----------------------')
    result = df.groupBy('Contract').pivot('Type').sum('TotalDuration').fillna(0)
    print('Pivot Type')
    print('-----------------------')
    # Caculate the most watch
    columns_to_compare = ['Television', 'Feature film', 
'Entertainment','Kid','Sport']
    result = result.withColumn('most_watch', greatest(*columns_to_compare))
    conditions = [when(col('most_watch') == col(c), c) for c in
columns_to_compare]
    result = result.withColumn('most_watch', coalesce(*conditions))
    print('Caculating the most watch')
    print('-----------------------')
    # Caculate the customer_taste
    conditions = [
        when(col('Entertainment') != 0, 'Entertainment'),
        when(col('Feature film') != 0, 'Feature film'),
        when(col('Kid') != 0, 'Kid'),
        when(col('Sport') != 0, 'Sport'),
        when(col('Television') != 0, 'Television')]
    result = result.withColumn('customer_taste', concat_ws('-', *conditions))
    print('Caculating the customer taste')
    print('-----------------------')
    # Caculate the customer activeness
    log_counts = df.groupBy('Contract').agg(count('Contract').alias('Log_count'))
    result = result.join(log_counts, 'Contract', 'left') \
    .withColumn('customer_activeness', when(col('Log_count') > 4,
'high').otherwise('low'))
    return(result)

# Loading Data to CSV
def data_load(result,save_path):
    print('Saving result output')
    print('-----------------------')
    result.write.csv(save_path,header = True)

# Importing ETL data to MySQL
def import_to_mysql(result):
    from pyspark.sql.types import StructType
 # Flatten struct columns if they exist
    for field in result.schema.fields:
        if isinstance(field.dataType, StructType):
            # Lấy tất cả các trường con từ struct
            for subfield in field.dataType.fields:
                column_name = f"{field.name}_{subfield.name}"
                result = result.withColumn(column_name,
col(f"{field.name}.{subfield.name}"))
            # Drop cột struct gốc
            result = result.drop(field.name)
    # MySQL connection details
    user = 'root'  # Replace with your MySQL username
    password = ''  # Replace with your MySQL password
    host = 'localhost'  # Your MySQL host
    port = '3306'  # Default MySQL port
    database = 'customer360_pipeline'  # The database name
    table = 'LogContent_ETL_data'  # Table name
    # MySQL connection properties
    mysql_properties = {
        'driver': 'com.mysql.cj.jdbc.Driver',
        'user': user,
        'password': password,
        'url': f'jdbc:mysql://{host}:{port}/{database}'}
    # Write DataFrame to MySQL
    result.write \
        .mode('overwrite') \
        .format('jdbc') \
        .option('driver', mysql_properties['driver']) \
        .option('url', mysql_properties['url']) \
        .option('dbtable', table) \
        .option('user', mysql_properties['user']) \
        .option('password', mysql_properties['password']) \
        .save()

# ETL Process
def main_task(start_date,end_date,path,save_path):
    print('----------------------')
    print('Transforming date')
    print('----------------------')
    day_list = date_transform(start_date,end_date)
    print('Transforming date completely')
    print('----------------------')
    print('Extracting data')
    print('----------------------')
    df = data_extract(day_list,path)
    print('Extracting data completely')
    print('----------------------')
    print('Transforming data')
    print('----------------------')
    result = data_transform(df)
    print('Transforming data completely')
    print('----------------------')
    print('Showing data sample')
    print('----------------------')
    result.show(20)
    print('----------------------')
    print('Loading data to CSV file')
    print('----------------------')
    data_load(result,save_path)
    print('Loading data to CSV file completely')
    print('----------------------')
    import_to_mysql(result)
    print('Importing data to MySQL completely')
    print('----------------------')
    return print('Task run successfully')

# Enter 'Path' containing data folder
path = 'C:\\Nguyễn Minh Khôi - EEC01\\study_data_DE\\Big Data\\CLass 4 - ETL
Pipeline\\log_content(short)\\'
# Enter start date and end date by according to syntax day = {yyyymmdd}
start_date = '20220401'
end_date = '20220430'
# Enter 'save_path' storing data passed ETL process
save_path = 'C:\\Nguyễn Minh Khôi - EEC01\\study_data_DE\\Big Data\\CLass 4 - ETL
Pipeline\\ETL_LogContent\\Clean_data.csv'
```

6.2. Finding keyword most search and mapping with the category

```
import findspark
findspark.init()
from pyspark.sql import SparkSession
from pyspark.sql.functions import *
from datetime import datetime
from datetime import date, timedelta
import os

spark = SparkSession.builder.getOrCreate()

# Processing the entered time period
def date_transform(start_date_6,end_date_6,start_date_7,end_date_7):
    start_date_6_dt = datetime.strptime(start_date_6, '%Y%m%d')
    end_date_6_dt = datetime.strptime(end_date_6,'%Y%m%d')
    day_list_6 = [(start_date_6_dt + timedelta(days=i)).strftime('%Y%m%d') for i
in range((end_date_6_dt - start_date_6_dt).days + 1)]
    start_date_7_dt = datetime.strptime(start_date_7, '%Y%m%d')
    end_date_7_dt = datetime.strptime(end_date_7,'%Y%m%d')
    day_list_7 = [(start_date_7_dt + timedelta(days=i)).strftime('%Y%m%d') for i
in range((end_date_7_dt - start_date_7_dt).days + 1)]
    day_list = day_list_6 + day_list_7
    return(day_list)

# Loading data based on the request time series
def data_extract(day_list,path):
    list_folder = os.listdir(path)
    folder_path = path + list_folder[0]
    parquet_files = [os.path.join(folder_path, f) for f in
os.listdir(folder_path) if f.endswith('.parquet')]
    df = spark.read.parquet(parquet_files[0])
    df = df.withColumn('Date', lit(datetime.strptime(day_list[0], '%Y%m%d')))
    for i in list_folder[1:]:
        folder_path = path + i
        parquet_files = [os.path.join(folder_path, f) for f in
os.listdir(folder_path) if f.endswith('.parquet')]
        data = spark.read.parquet(parquet_files[0])
        data = data.withColumn('Date', lit(datetime.strptime(i, '%Y%m%d')))
        df = df.union(data)
    return(df)

# Summary of the most searched keywords
def keyword_most_search(df):
    df = df.withColumn('keyword', lower(df['keyword']))
    keywork_most_search =
df.groupBy('keyword','category').agg(count('keyword').alias('count'))
    keywork_most_search =  keywork_most_search.orderBy(col('count').desc())
    print('Loading data to CSV')
    print('----------------------')
    keywork_most_search.write.csv('C:\\Nguyễn Minh Khôi -
EEC01\\study_data_DE\\Big Data\\Class 7 - Final Project\\keyword_most_search')
    return(keywork_most_search)

def import_to_mysql(result):
    from pyspark.sql.types import StructType
 # Flatten struct columns if they exist
    for field in result.schema.fields:
        if isinstance(field.dataType, StructType):
            # Lấy tất cả các trường con từ struct
            for subfield in field.dataType.fields:
                column_name = f"{field.name}_{subfield.name}"
                result = result.withColumn(column_name,
col(f"{field.name}.{subfield.name}"))
            # Drop cột struct gốc
            result = result.drop(field.name)
    # MySQL connection details
    user = 'root'  # Replace with your MySQL username
    password = ''  # Replace with your MySQL password
    host = 'localhost'  # Your MySQL host
    port = '3306'  # Default MySQL port
    database = 'customer360_pipeline'  # The database name
    table = 'keyword_most_search'  # Table name
    # MySQL connection properties
    mysql_properties = {
        'driver': 'com.mysql.cj.jdbc.Driver',
        'user': user,
        'password': password,
        'url': f'jdbc:mysql://{host}:{port}/{database}'}
    # Write DataFrame to MySQL
    result.write \
        .mode('overwrite') \
        .format('jdbc') \
        .option('driver', mysql_properties['driver']) \
        .option('url', mysql_properties['url']) \
        .option('dbtable', table) \
        .option('user', mysql_properties['user']) \
        .option('password', mysql_properties['password']) \
        .save()
    return print('Loading data to MySQL completely')

# Main Task
def main_task(path,start_date_6,end_date_6,start_date_7,end_date_7):
    print('----------------------')
    print('Transforming date')
    day_list = date_transform(start_date_6,end_date_6,start_date_7,end_date_7)
    print('----------------------')
    print('Transforming date completely')
    print('----------------------')
    print('Extracting data')
    print('----------------------')
    df = data_extract(day_list,path)
    print('Extracting data completely')
    print('----------------------')
    print('Caculating the keyword most search')
    print('----------------------')
    result = keyword_most_search(df)
    result.show(30)
    print('Caculating the keyword most search comletely')
    print('----------------------')
    print('Loading data to MySQL')
    print('----------------------')
    import_to_mysql(result)
    print('----------------------')
    return print('Task run successfully')


# Enter the path containing data
path = 'C:\\Nguyễn Minh Khôi - EEC01\\study_data_DE\\Big Data\\Class 7 - Final 
Project\\log_search\\'
# Enter the start date in June
start_date_6 = '20220601'
# Enter the end date in June
end_date_6 = '20220614'
# Enter the start date in July
start_date_7 = '20220701'
# Enter the end date in July
```

```
end_date_7 = '20220714'
```

6.3. Customer trending

```
import findspark
findspark.init()
from pyspark.sql import SparkSession
from pyspark.sql.functions import *
from datetime import datetime
from datetime import date, timedelta
import os
from pyspark.sql.window import *

spark = SparkSession.builder.getOrCreate()

# Processing the entered time period
def date_transform(start_date_6,end_date_6,start_date_7,end_date_7):
    start_date_6_dt = datetime.strptime(start_date_6, '%Y%m%d')
    end_date_6_dt = datetime.strptime(end_date_6,'%Y%m%d')
    day_list_6 = [(start_date_6_dt + timedelta(days=i)).strftime('%Y%m%d') for i
in range((end_date_6_dt - start_date_6_dt).days + 1)]
    start_date_7_dt = datetime.strptime(start_date_7, '%Y%m%d')
    end_date_7_dt = datetime.strptime(end_date_7,'%Y%m%d')
    day_list_7 = [(start_date_7_dt + timedelta(days=i)).strftime('%Y%m%d') for i
in range((end_date_7_dt - start_date_7_dt).days + 1)]
    day_list = day_list_6 + day_list_7
    return(day_list)

# Loading data based on the request time series
def data_extract(day_list,path):
    list_folder = os.listdir(path)
    folder_path = path + list_folder[0]
    parquet_files = [os.path.join(folder_path, f) for f in
os.listdir(folder_path) if f.endswith('.parquet')]
    df = spark.read.parquet(parquet_files[0])
    df = df.withColumn('Date', lit(datetime.strptime(day_list[0], '%Y%m%d')))
    for i in list_folder[1:]:
        folder_path = path + i
        parquet_files = [os.path.join(folder_path, f) for f in
os.listdir(folder_path) if f.endswith('.parquet')]
        data = spark.read.parquet(parquet_files[0])
        data = data.withColumn('Date', lit(datetime.strptime(i, '%Y%m%d')))
        df = df.union(data)
    return(df)

# Read the Category file by keywords
```

```
def read_category_file(category_path):
    category = spark.read.csv(category_path, header = True)
    return(category)

# Calculate and compare access information for each user in June and July
def compare_June_and_July(df, category):
    df = df.filter(col('category') == 'enter').select('user_id','keyword','Date')
    # Create a 'month' column based on the values in the 'Date' column
    df = df.withColumn('month', month('Date'))
    # Calculate the total number of access grouped by 'keyword', 'user_id', and
'month'
    df =
df.groupBy('keyword','user_id','month').agg(count('keyword').alias('count'))
    df = df.join(category, on = 'keyword').drop('No')
    # Rank the keyword access by each user and month
    df = df.withColumn('rank',
row_number().over(Window.partitionBy('user_id','month').orderBy(col('count').desc
())))
    df = df.filter(col('rank') == '1')
    # Filter information for June
    data_month6 = df.filter(col('month') == '6').withColumnRenamed('keyword',
'most_search_month6') \
        .withColumnRenamed('Category', 'category_month6')
    data_month6 = data_month6.select('user_id', 'most_search_month6',
'category_month6')
    # Filter information for July
    data_month7 = df.filter(col('month') == '7').withColumnRenamed('keyword',
'most_search_month7') \
        .withColumnRenamed('Category', 'category_month7')
    data_month7 = data_month7.select('user_id', 'most_search_month7',
'category_month7')
    # Join 2 tables of data for June and July
    data = data_month6.join(data_month7, on = 'user_id')
    # Calculate Trending_type to see if the most searched keywords of users
changed between June and July
    data = data.withColumn('Trending_type',
                       when((col('category_month6') == col('category_month7')),
'Unchanged')
                       .otherwise('Changed'))
    # Calculate Previous to understand the changes in user access
    data = data.withColumn('Previous',
                       when((col('category_month6') == col('category_month7')),
'Unchanged')
                       .otherwise(concat(col('category_month6'), lit('-
'),col('category_month7'))))
    return(data)
```

```
def import_to_mysql(data):
    from pyspark.sql.types import StructType
 # Flatten struct columns if they exist
    for field in data.schema.fields:
        if isinstance(field.dataType, StructType):
            # Lấy tất cả các trường con từ struct
            for subfield in field.dataType.fields:
                column_name = f"{field.name}_{subfield.name}"
                data = data.withColumn(column_name,
col(f"{field.name}.{subfield.name}"))
            # Drop cột struct gốc
            data = data.drop(field.name)
    # MySQL connection details
    user = 'root'  # Replace with your MySQL username
    password = ''  # Replace with your MySQL password
    host = 'localhost'  # Your MySQL host
    port = '3306'  # Default MySQL port
    database = 'customer360_pipeline'  # The database name
    table = 'LogSearch_Data_Final'  # Table name
    # MySQL connection properties
    mysql_properties = {
        'driver': 'com.mysql.cj.jdbc.Driver',
        'user': user,
        'password': password,
        'url': f'jdbc:mysql://{host}:{port}/{database}'}
    # Write DataFrame to MySQL
    data.write \
        .mode('overwrite') \
        .format('jdbc') \
        .option('driver', mysql_properties['driver']) \
        .option('url', mysql_properties['url']) \
        .option('dbtable', table) \
        .option('user', mysql_properties['user']) \
        .option('password', mysql_properties['password']) \
        .save()
    return print('Loading data to MySQL completely')

# Main Task
def
main_task(path,start_date_6,end_date_6,start_date_7,end_date_7,category_path):
    print('-----------------------')
    print('Transforming date')
    day_list = date_transform(start_date_6,end_date_6,start_date_7,end_date_7)
    print('-----------------------')
    print('Transforming date completely')
    print('----------------------')
    print('Extracting data')
    print('----------------------')
    df = data_extract(day_list,path)
    df.show(30)
    print('Extracting data completely')
    print('----------------------')
    print('Read the Category file by keywords')
    print('----------------------')
    category = read_category_file(category_path)
    category.show(30)
    print('Read the Category file by keywords comletely')
    print('----------------------')
    print('Compare access information for each user in June and July')
    print('----------------------')
    data = compare_June_and_July(df, category)
    data.show(30)
    print('Compare access information for each user in June and July completely')
    print('----------------------')
    print('Loading data to MySQL')
    print('----------------------')
    import_to_mysql(data)
    print('----------------------')
    return print('Task run successfully')

# Enter the path containing data
path = 'C:\\Nguyễn Minh Khôi - EEC01\\study_data_DE\\Big Data\\Class 7 - Final
Project\\log_search\\'
# Enter the path containing category data
category_path = "C:\\Nguyễn Minh Khôi - EEC01\\study_data_DE\\Big Data\\Class 7 -
Final Project\\keyword_category.csv"
# Enter the start date in June
start_date_6 = '20220601'
# Enter the end date in June
end_date_6 = '20220614'
# Enter the start date in July
start_date_7 = '20220701'
# Enter the end date in July
end_date_7 = '20220714'
```

--- THE END ---