

Name: Abhishek Obla Hema
Email: oh_abhishek@yahoo.com

Spark Assignment-1(Marketing Campaign Analysis) Documentation

This file details and describes all the attached files for the Spark Assignment- 1

Tools Used:

1. Python3 – Microsoft VScode
2. Apache Spark (On GCP Cluster)
3. GCP services – DataProc/GCS
4. Jupyter Lab
5. Apache Hive

Files Attached:

1. Spark_Ass1.pdf – This file
2. Spark_Marketingdata– Pyspark File that details all the analysis
3. HQL_datastore.txt – HQL commands used to create external tables off the data

Process and File Descriptions:

Step 1:

First I created three dataframes:

- Campaigns
- Users
- Stores

```
[5]: # Show the dataframes
df_campaigns.show(5)
df_users.show(5)
df_stores.show(5)
```

campaign_id	campaign_name	campaign_country	os_type	device_type	place_id	user_id	event_type	event_time
ABCDFAE	Food category tar...	USA	ios	apple	CASSBB-11	1264374214654454321	impression	2018-10-12 13:10:05
ABCDFAE	Food category tar...	USA	android	MOTOROLA	CADGBD-13	1674374214654454321	impression	2018-10-12 13:09:04
ABCDFAE	Food category tar...	USA	android	SAMSUNG	BADGBA-12	5747421465445443	video ad	2018-10-12 13:10:10
ABCDFAE	Food category tar...	USA	android	SAMSUNG	CASSBB-11	1864374214654454132	click	2018-10-12 13:10:12

user_id	country	gender	age_group	category
1264374214654454321	USA	male	18-25	[shopper, student]
1674374214654454321	USA	female	25-50	[parent]
5747421465445443	USA	male	25-50	[shopper, parent,...]
1864374214654454132	USA	male	50+	[professional]
14537421465445443	USA	female	18-25	[shopper, student]

only showing top 5 rows

store_name	place_ids
McDonald	[CASSBB-11, CADGB...
BurgerKing	[CASSBB-11]
Macy's	[BADGBA-13, CASSB...
shoppers stop	[BADGBA-12]

Step 2:

I then performed the necessary transformations that has been asked in the question and placed the output in separate directories in the HDFS .

```
[6]: # Extract date and hour from the event_time column
df_campaigns = df_campaigns.withColumn("event_time", F.col("event_time").cast("timestamp"))
df_campaigns = df_campaigns.withColumn("date", F.to_date("event_time"))
df_campaigns = df_campaigns.withColumn("hour", F.hour("event_time"))

[7]: # Define the output path
hdfs_output_path1 = '/tmp/marketing_data/output/'

# Q1. Analyze data for each campaign_id, date, hour, os_type & value to get all the events with counts
result_q1 = (
    df_campaigns.groupBy("campaign_id", "date", "hour", "os_type", "event_type")
    .agg(F.count("event_type").alias("event_count"))
    .groupBy("campaign_id", "date", "hour", "os_type")
    .pivot("event_type")
    .agg(F.first("event_count"))
    .fillna(0)
    .select(
        "campaign_id",
        "date",
        "hour",
        "os_type",
        F.struct(
            F.col("impression").alias("impression"),
            F.col("click").alias("click"),
            F.col("video_ad").alias("video_ad"),
        ).alias("event"),
    )
)

[8]: result_q1.show()

[Stage 14:>                                     (0 + 1) / 1]
+-----+-----+-----+-----+-----+
|campaign_id|   date|hour|os_type|   event|
+-----+-----+-----+-----+-----+
|   ABCDFAE|2018-10-12|  13|android|{1, 1, 1}|
|   ABCDFAE|2018-10-12|  13|   ios|{1, 0, 0}|
+-----+-----+-----+-----+-----+

[9]: # Save the result to HDFS
result_q1.write.json(hdfs_output_path1 + "q1_output", mode="overwrite")
```

```
[14]: # Define the output path
hdfs_output_path2 = '/tmp/marketing_data/output2/'

# Q2. Analyze data for each campaign_id, date, hour, store_name & value to get all the events with counts
result_q2 = (
    df_campaigns.join(df_stores, F.array_contains(df_stores.place_ids, df_campaigns.place_id), "inner")
    .groupBy("campaign_id", "date", "hour", "store_name", "event_type")
    .agg(F.count("event_type").alias("event_count"))
    .groupBy("campaign_id", "date", "hour", "store_name")
    .pivot("event_type")
    .agg(F.first("event_count"))
    .fillna(0)
    .select(
        "campaign_id",
        "date",
        "hour",
        "store_name",
        F.struct(
            F.col("impression").alias("impression"),
            F.col("click").alias("click"),
            F.col("video_ad").alias("video_ad"),
        ).alias("event"),
    )
)
```

```
[15]: result_q2.show()
```

campaign_id	date	hour	store_name	event
ABCDFAE	2018-10-12	13	BurgerKing	{1, 1, 0}
ABCDFAE	2018-10-12	13	McDonald	{2, 1, 0}
ABCDFAE	2018-10-12	13	shoppers stop	{0, 0, 1}

```
[16]: # Save the result to HDFS
result_q2.write.json(hdfs_output_path2 + "q2_output", mode="overwrite")
```

```
[18]: # Define the output path
hdfs_output_path3 = '/tmp/marketing_data/output3/'

# Q3. Analyze data for each campaign_id, date, hour, gender_type & value to get all the events with counts
result_q3 = (
    df_campaigns.join(df_users, "user_id", "inner")
    .groupBy("campaign_id", "date", "hour", "gender", "event_type")
    .agg(F.count("event_type").alias("event_count"))
    .groupBy("campaign_id", "date", "hour", "gender")
    .pivot("event_type")
    .agg(F.first("event_count"))
    .fillna(0)
    .select(
        "campaign_id",
        "date",
        "hour",
        "gender",
        F.struct(
            F.col("impression").alias("impression"),
            F.col("click").alias("click"),
            F.col("video_ad").alias("video_ad"),
        ).alias("event"),
    )
)
```

```
[19]: result_q3.show()
```

campaign_id	date	hour	gender	event
ABCDFAE	2018-10-12	13	male	{1, 1, 1}
ABCDFAE	2018-10-12	13	female	{1, 0, 0}

```
[20]: # Save the result to HDFS
result_q3.write.json(hdfs_output_path3 + "q3_output", mode="overwrite")
```

Step 3:

We can check the output has been saved using the UI

Browse Directory

Show 25 entries

Search:

<input type="checkbox"/>	Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name	
<input type="checkbox"/>	-rW-r--r--	absf1r3	hadoop	1.25 KB	Dec 15 04:00	2	128 MB	ad_campaigns_data.json	
<input type="checkbox"/>	drwxr-xr-x	root	hadoop	0 B	Dec 15 05:19	0	0 B	output	
<input type="checkbox"/>	drwxr-xr-x	root	hadoop	0 B	Dec 15 05:23	0	0 B	output2	
<input type="checkbox"/>	drwxr-xr-x	root	hadoop	0 B	Dec 15 05:29	0	0 B	output3	
<input type="checkbox"/>	-rW-r--r--	absf1r3	hadoop	471 B	Dec 15 04:00	2	128 MB	store_data.json	
<input type="checkbox"/>	-rW-r--r--	absf1r3	hadoop	1.13 KB	Dec 15 04:00	2	128 MB	user_profile_data.json	

Showing 1 to 6 of 6 entries

Previous

1

Next

Hadoop, 2023.

Step 4:

Using HQL statements I then created Hive external tables using JSON serde.

```
CREATE EXTERNAL TABLE IF NOT EXISTS q1_output (  
  campaign_id STRING,  
  `date` STRING,  
  hour INT,  
  os_type STRING,  
  impression INT,  
  click INT,  
  video_ad INT  
)  
ROW FORMAT SERDE 'org.apache.hive.hcatalog.data.JsonSerDe'  
WITH SERDEPROPERTIES (  
  'serialization.format' = '1'  
)  
LOCATION '/tmp/marketing_data/output/q1_output';  
  
CREATE EXTERNAL TABLE IF NOT EXISTS q2_output (  
  campaign_id STRING,  
  `date` STRING,  
  hour INT,  
  store_name STRING,  
  event STRUCT<impression: INT, click: INT, video_ad: INT>  
)  
ROW FORMAT SERDE 'org.apache.hive.hcatalog.data.JsonSerDe'  
WITH SERDEPROPERTIES (  
  'ignore.malformed.json' = 'true'  
)  
STORED AS TEXTFILE  
LOCATION '/tmp/marketing_data/output2/q2_output';  
  
CREATE EXTERNAL TABLE IF NOT EXISTS q3_output (  
  campaign_id STRING,  
  `date` STRING,  
  hour INT,  
  gender STRING,  
  event STRUCT<impression: INT, click: INT, video_ad: INT>  
)  
ROW FORMAT SERDE 'org.apache.hive.hcatalog.data.JsonSerDe'  
WITH SERDEPROPERTIES (  
  'ignore.malformed.json' = 'true'  
)  
STORED AS TEXTFILE  
LOCATION '/tmp/marketing_data/output3/q3_output';
```

Step 5:

Querying Hive tables to check the output data to make sure

```
hive> show databases;
OK
airflow
default
hive_db
market_data
tables_by_spark
Time taken: 0.734 seconds, Fetched: 5 row(s)
hive> use market_data;
OK
Time taken: 0.122 seconds
hive> show tables;
OK
q1_output
q2_output
q3_output
Time taken: 0.08 seconds, Fetched: 3 row(s)
hive> Select * from q1_output;
OK
ABCDFAE 2018-10-12      13      android {"impression":1,"click":1,"video_ad":1}
ABCDFAE 2018-10-12      13      ios      {"impression":1,"click":0,"video_ad":0}
Time taken: 3.283 seconds, Fetched: 2 row(s)
hive> Select * from q2_output;
OK
ABCDFAE 2018-10-12      13      BurgerKing {"impression":1,"click":1,"video_ad":0}
ABCDFAE 2018-10-12      13      McDonald {"impression":2,"click":1,"video_ad":0}
ABCDFAE 2018-10-12      13      shoppers stop {"impression":0,"click":0,"video_ad":1}
Time taken: 0.39 seconds, Fetched: 3 row(s)
hive> Select * from q3_output;
OK
ABCDFAE 2018-10-12      13      male      {"impression":1,"click":1,"video_ad":1}
ABCDFAE 2018-10-12      13      female {"impression":1,"click":0,"video_ad":0}
Time taken: 0.311 seconds, Fetched: 2 row(s)
hive> █
```