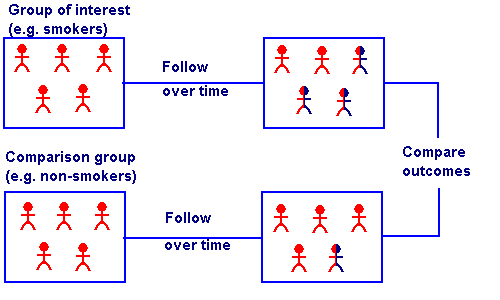
Cohort Analysis Using Apache Spark

**Note: This is just an experiment to figure out customers from which marketing medium like Google, Facebook, Email, SMS have more retention rate and less churn rate percentage.**

A “**C**ohort” is a group of people that share a certain characteristics. Usually, but not always. Practice of studying the activities or habits of specific cohorts over a set period of time is called Cohort Analysis. Usually, Most marketers prefer a month by month analysis over others. But, It can be performed week or day level as well. I choose week as time metric for this experiment.

https://miro.medium.com/freeze/max/60/1*6PY5rmcVvE0ZiIArLXUx1A.gif?q=20



The Outcome of the analysis can be used to spot the patterns or changes in customer behaviour throughout their journey and why there is a change by comparing result with marketing events calendar. It is actually more of a prediction than analysis. E-commerce companies makes use Cohort Analysis to figure how the brand is performing over a period of time and which Cohort (Age 25 -30 group or Bangalore city customer) group are most valuable to their brand.

How Cohort Growth metrics stands out from normal Vanity metrics like number of new sessions is that, it will help marketers make business decisions. Problem with Vanity metrics is not just about waste of time analyzing it. It can actually mislead business decisions sometimes. For instance, If the number of new sessions goes up, Is it good that new users visit your site? Or Is it a bad thing that returning users retention rate is going down?

A million guys walk into a Sillicon Valley bar. None of them buy anything. The bar is declared a rousing success. — Anonymous, Quora

## **Apache Spark:**

Apache Spark is a Unified analytics engine for large-scale data processing. Runs roughly 10X to 100X faster than a Hadoop MapReduce because of In-memory computation and other spark’s core components like Catalyst Query Optimizer, **D**irected **A**cyclic **G**raph scheduler and its Physical Execution Engine. I will be discussing about it in detail in my upcoming blogs. In the meantime, don’t hesitate to refer the documentation for each [[1]](https://databricks.com/blog/2015/04/13/deep-dive-into-spark-sqls-catalyst-optimizer.html?source=post_page---------------------------), [[2]](https://data-flair.training/blogs/dag-in-apache-spark/?source=post_page---------------------------), [3].

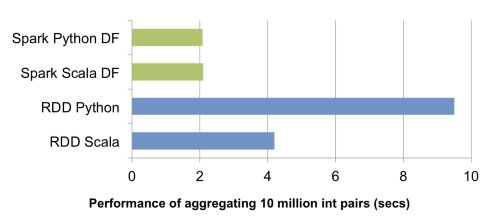
There are two core abstractions in spark needs to be discussed before diving into an implementation aspect of this analysis. One is **R**esilient **D**istributed **D**atasets (RDD) and another one is Data Frames.

## **R**esilient **D**istributed **D**atasets (RDD):

A Resilient Distributed Dataset (RDD), the basic abstraction in Spark. Represents an immutable, partitioned collection of elements that can be operated on in parallel. These entities exist in memory and by their very nature are immutable. Due to its immutability nature, new RDD will be created when some transformation applied to it. If there is any failure in transformation or action on RDD during the execution phase, It is possible to roll back to previous functioning state and helps to perform the same action in a different node to achieve fault tolerance.

## **Data Frames:**

https://miro.medium.com/max/60/1*Tu0y1tPPOPQ14Ww14odO4Q.png?q=20



RDD by nature don’t have schema attached to it, but they can be extended with help of DataFrames. Data Frames contain schema functionality to the dataset it contains, which helps in handling structured data set. It is relatively faster then RDD since it has more information about dataset it holds. Let’s get into action now.

I had exported six weeks worth of signed-in users events which comes roughly around two million records from our events database to perform cohort analysis on it to figure out which marketing medium performs better in terms of retaining return users.

userId,eventType,productId,eventTime  
26309c83–19e7, WISHLIST, 184061,1527813154155  
fc07ec53-ebfe, PRODUCT\_VIEW, 162654,1527813154200  
82b73ce3-e5c0, PRODUCT\_VIEW, 183104,1527813163406  
26309c83–19e7, WISHLIST, 184061,1527813169063  
82b73ce3-e5c0, PRODUCT\_VIEW, 183104,1527813176444  
2d1c7da8–6967, PRODUCT\_VIEW, 184746,1527813186410  
26309c83–19e7, WISHLIST, 184061,1527813187331  
20d41d6b-51dc, UTM\_PARAMS, utm\_source:sms,1527837139873

## Creating Spark Session and Data Frame:

Spark Session provides a single point of entry to interact with the underlying Spark functionality and allows programming Spark with DataFrame and Dataset APIs. In order to use APIs of SQL, HIVE, and Streaming, no need to create separate contexts as Spark Session includes all the APIs.

spark = SparkSession.builder.  
 appName("cohort").  
 config("spark.some.config.option","cohort").  
 getOrCreate()  
df = spark.read.format(“csv”).  
 option(“header”, “true”).load(file\_path)

**Join Product View Events and Marketing Events:**

To get customers who are all viewed product from marketing medium, I’ve created two Data frames using below, one to store View actions and another one to store UTM source events. By joining both data frames using common user Ids, We created new Data Frame which now contains all product view events with a corresponding marketing medium source.

track\_events = df.where(df["eventType"] == "UTM\_PARAMS")  
view\_events = df.where(df["eventType"] == "PRODUCT\_VIEW")  
df1 = track\_events.alias('df1')  
df2 = view\_events.alias('df2')  
track\_plus\_view\_events = df1.join(df2, df1.userId == df2.userId).select(['df1.productId', 'df2.userId', 'df2.eventTime'])cohort = track\_plus\_view\_events.rdd.map((lambda x: (removeUtmParam(x["productId"]), x["userId"], getCohortGroupId(int(x["eventTime"]))).filter((lambda x: "utm\_source" in x["productId"]))))

Spark does something called Lazy Evaluation, Which means all transformation will be applied only when some actions like Show(), take(), count() , write() triggered on RDD. Till then, DAG scheduler will keep working on optimizing what job it needs to execute to reach the action it has been asked to do. If you look at the last line of the above code snippets, map is applied first and then filter. But DAG scheduler in Spark is smart enough to apply filter first on RDD to reduce the number of records to be processed for mapping.

## Partitioning Cohorts:

Time to create Cohorts based on Marketing medium sources like Google, Facebook, SMS, email.

cohort\_df = cohort.toDF(["source", "userId", "cohortId"])  
cohort\_df.write.partitionBy("source").csv("cohorts")

When write action is triggered, all the transformation applied on RDD will be processed as per logical execution plan. Once it is partitioned by source, we can see a different folder for each source like below. Once all sources are partitioned like below, we can divide each source by week by week and just compare who are in the first week are still returning back in an upcoming week.

https://miro.medium.com/max/60/1*MGu7_dRQJSIalqhimIf3BQ.png?q=20



## Cohort Result table:

## Some Key Take Aways:

1. Unsurprisingly, there is a big drop in user retention of the SMS medium from second week itself. Considering user mostly won’t usually sign in on the mobile website.
2. Churn rate from Email and Criteo medium is gradually reducing unlike others. Which is considerably better compared to other mediums.