## **Final Project Report**

*Case Study in Web analytics*

**Problem Statement**

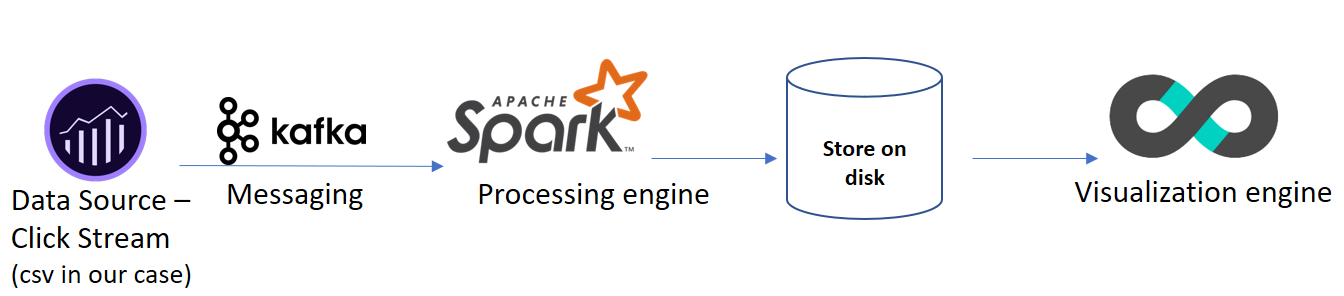
To analyze online traffic of a website, which would help the e-commerce division of that organization to understand the website traffic numbers, understand customer demographics. The tool can also act as a critical tool for predicting business trends and as an important means for online market research.

For creation of this tool, I will be using Apache Kafka as messaging system between producers and consumers, use Apache Spark for ETL & analytics, and finally use Apache Superset for

**real-time data visualization.**

To answer the questions above I would be analyzing on clickstream data - usually in industry to generate clickstream data – Google Tag Manager or Adobe Dynamic Tag manager are used and are implemented on ecommerce websites

In my project – I had scraped data out of web and stored in form of csv files with all appropriate masking *(to hide any kind of PII)* – I will be writing a producer job which would do simulation of clickstream events



*The above picture shows the overall flow of the project – and the order how clickstream data will be processed, and all the tools used at different stages of processing.*

By implementing the above flow, I will try to answer following business questions:

* How many orders does the website receive per minute by each platform (mobile vs desktop)
* What is the most clicked element on the website - Knowing this can help the business to optimize their website design and pages
* What time does the website traffic is highest – Knowing this can help the business to plan their infrastructure accordingly

## **Description of Data**

The data was created through Adobe Debugger tool *(Data generation methodology described in appendix)*

This dataset contains summary information of clickstream data, which is present around world wide web. There are usually 161 columns in this data source.

*(For more info see: data\_dictonary.txt file)*

For demo purposes, I have reduced the dimension of this data and masked all the information (PII – if present to hide user identity)

**File Name: orders\_data.csv**

**File Size: 151 MB**

**File Format: Comma Separated Values**

**Download Link:** [Link to Dataset](https://drive.google.com/drive/folders/1mCvvpUMkaiqGDL8xDmzEFibWMat4iwk4?usp=sharing)

This file contains the following columns:

* visitorid: *Unique visitorid assigned to everyone on web*
* sessionid: *Unique session assigned to each visitor*
* page\_name: *Name of Page on which visitor is currently*
* device:  *Platform through which customer is accessing website*
* event: *Button interacted by the visitor during session*
* date\_time: *Timestamp of that interaction*

**File Name: clicks\_data.csv (Sample)**

**File Size: 10 MB**

**File Format: Comma Separated Values**

**Download Link:** [Link to Dataset](https://drive.google.com/drive/folders/1mCvvpUMkaiqGDL8xDmzEFibWMat4iwk4?usp=sharing)

This file contains the following columns:

* visitorid: *Unique visitorid assigned to everyone on web*
* sessionid: *Unique session assigned to each visitor*
* page\_name: *Name of Page on which visitor is currently*
* device:  *Platform through which customer is accessing website*
* event: *Button interacted by the visitor during session*
* date\_time: *Timestamp of that interaction*

## **Description of Hardware**

I developed and executed this application on a HP EliteBook laptop running Windows 10 Enterprise edition (64-bit) on an Intel® Core™ i5-6200 CPU processor. The laptop is equipped with 16 gigabytes of RAM and no dedicated graphics processing unit.

The machine is also running a CentOS 7 on a virtual machine

## **Description of Software**

This project is entirely developed using the Python (version 2.7). I have been using Python on centos terminal and have been also using on gedit editor on CentOS

CentOS: [Download Link](https://www.centos.org/download/)

Python 2.7: [Download Link](https://www.python.org/downloads/)

gedit: [Download Link](https://gedit.en.softonic.com/)

For Exploratory data analysis also known as initial data analysis – I used Microsoft’s Power BI on Windows 10

Power BI: [Download Link](https://powerbi.microsoft.com/en-us/get-started/?&OCID=AID719832_SEM_M2kNt4pL&lnkd=Google_PowerBI_Brand&gclid=CjwKCAiA9K3gBRA4EiwACEhFe1RucWp5-Zj0XQPYpMwV0XPi53wBlxRb5oWUV6FR5aStEd0cvIjyoBoCFjYQAvD_BwE)

I used Apache Kafka for data streaming and Apache Spark for Data Processing

Apache Kafka 2.12:  [Download Link](https://kafka.apache.org/downloads)

Apache Spark 2.3.1:  [Download Link](https://spark.apache.org/downloads.html)

For Scheduling jobs, I used crontab - cronie package on CentOS

More info: [Link](https://www.rosehosting.com/blog/automate-system-tasks-using-cron-on-centos-7/)

For Real-time Data Visualization I used Apache Superset

Apache Superset: ([Apache Superset Link](https://superset.incubator.apache.org/installation.html))

## **Steps and Demonstration**

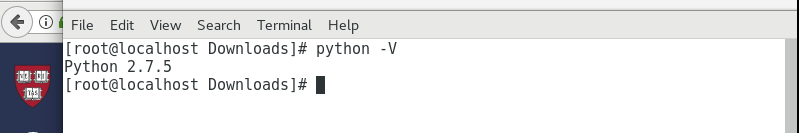
### **Installation**:

***Installing Python on CentOS***

Checking the version of python installed on the machine

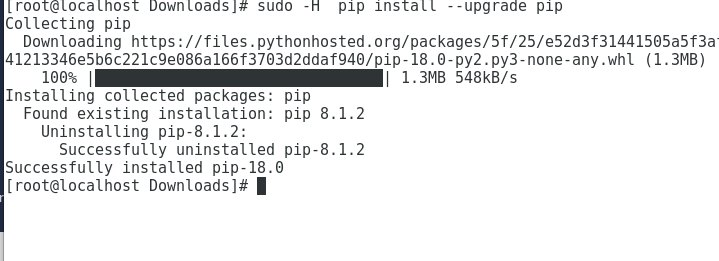
In my VM python was pre-installed, but to make sure every package is updated python I used python-pip later.

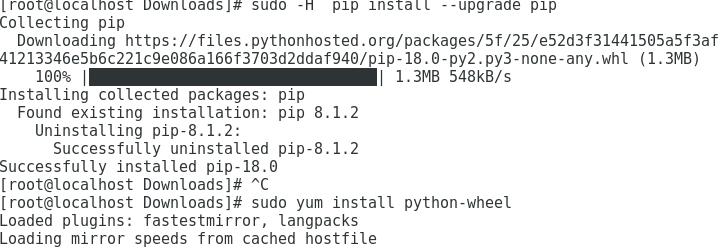
**Command:** **python -v**



***Installing Pip on CentOS – Through this we will install pyspark***

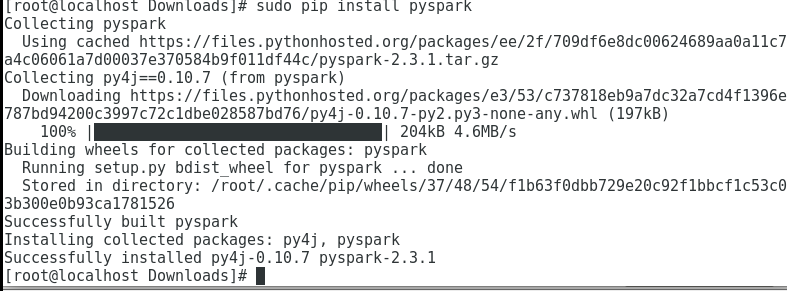
**Command:** **sudo -H pip install –upgrade pip**



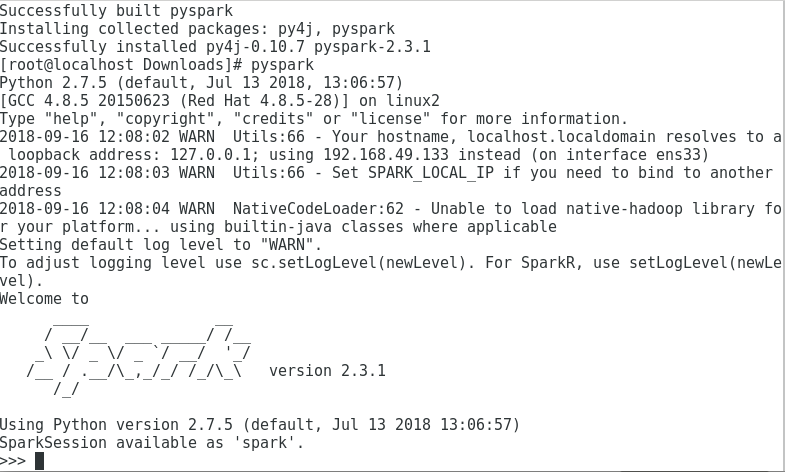


***Installing pyspark on the CENTOS VM.***

**Command:** **sudo pip install pyspark**



Opening pyspark shell successfully:



***Installing Kafka on the CENTOS VM.***

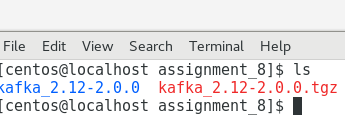
**Code:**

wget http://www-eu.apache.org/dist/kafka/2.0.0/kafka\_2.12-2.0.0.tgz

sudo cp kafka\_2.12-2.0.0 /opt

sudo mkdir /opt/kafka

sudo tar -xvf kafka\_2.12-2.0.0.tgz



Once, packages have been downloaded, I had to set up the home directories into bash profile

**Code:**

echo "export KAFKA\_HOME='/opt/kafka'" >> /home/centos/.bashrc

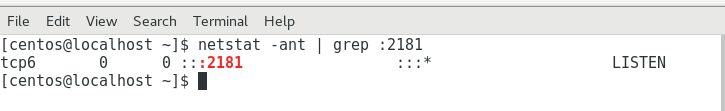
echo "export PATH=$PATH:$KAFKA\_HOME/bin" >> /home/centos/.bashrc

source /home/centos/.bashrc

Starting the Zookeeper services, on CENTOS VM

**Code:**

netstat -ant | grep :2181



As, we can see from above that at 2181 Port, tcp service is in listening mode. Before running Kafka Services, we need to run Zookeper services as well, which I will be showing when I run the topics (*messaging system*)

***Installing Apache Superset on the CENTOS VM.***

Detailed Instructions can be found [here](https://superset.incubator.apache.org/installation.html). The biggest challenge while using apache superset is maintaining Python 3.6+ environment Therefore, you would need to create a virtual environment with python 3.6

You will also have to ensure all the dependencies exist for building super-set

**Code:**

sudo yum upgrade python-setuptools

sudo yum install gcc gcc-c++ libffi-devel python-devel python-pip python-wheel openssl-devel libsasl2-devel openldap-devel

**Code:**

pip install virtualenv

Over here, I am running virtual environment under python 3.6 and naming it as new\_env

**Code:**

virtualenv -p python3.6 new\_env

cd new\_env/

source bin/activate

Then you need to use pip functionality to install superset

**Code:**

pip install superset

fabmanager create-admin --app superset **##Enter username and password**

superset db upgrade

superset init

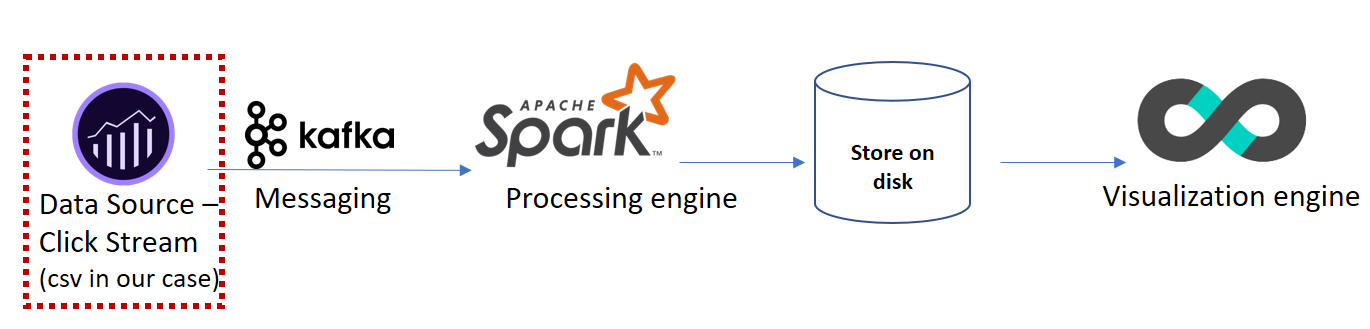
superset runserver -d

***Installing Power BI on the Windows machine.***

Power BI: [Download Link](https://powerbi.microsoft.com/en-us/get-started/?&OCID=AID719832_SEM_M2kNt4pL&lnkd=Google_PowerBI_Brand&gclid=CjwKCAiA9K3gBRA4EiwACEhFe1RucWp5-Zj0XQPYpMwV0XPi53wBlxRb5oWUV6FR5aStEd0cvIjyoBoCFjYQAvD_BwE)

Go to the link above and download the executable and run the executable on windows machine

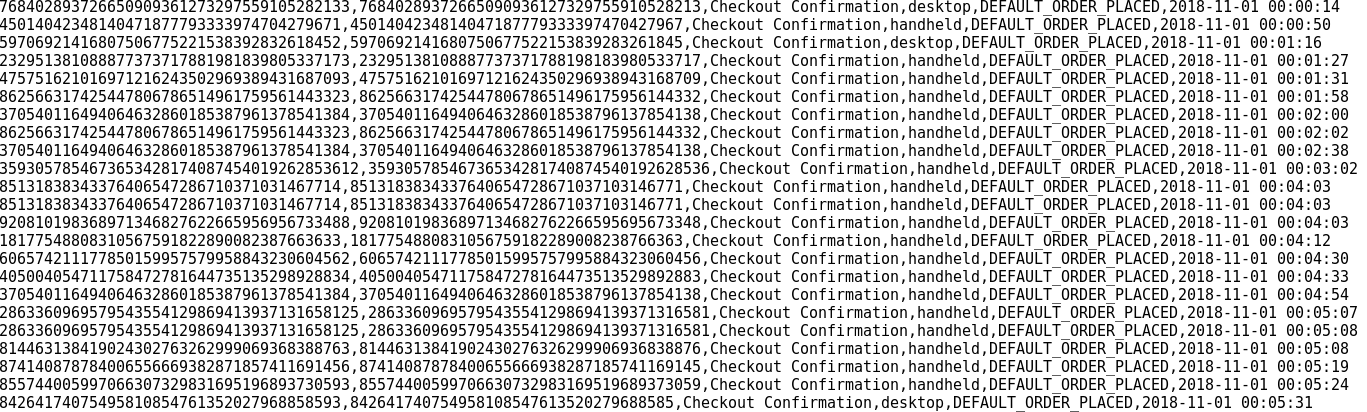
### **Data**:



So, I will be showing and be demonstrating the first part of the process.

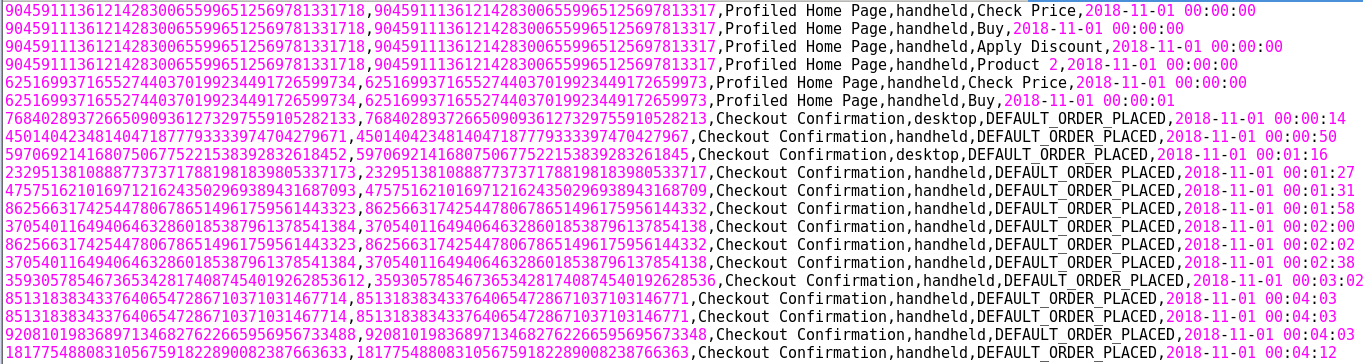
Firstly, we will be looking into orders\_data.csv and clicks\_data.csv file

The downloaded **orders\_data.csv** file looks as follows:



This data file is in the file is 150MB, this contains only order information of all the visitors coming to the website and placing an order. As you can see, that this data has a time-series component. So, I will be using this data to create real-time dashboard – Where I will be seeing at orders in real-time by each platform

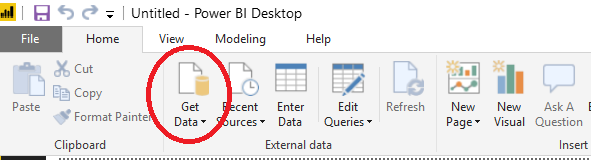
The downloaded **clicks\_data.csv** file looks as follows:



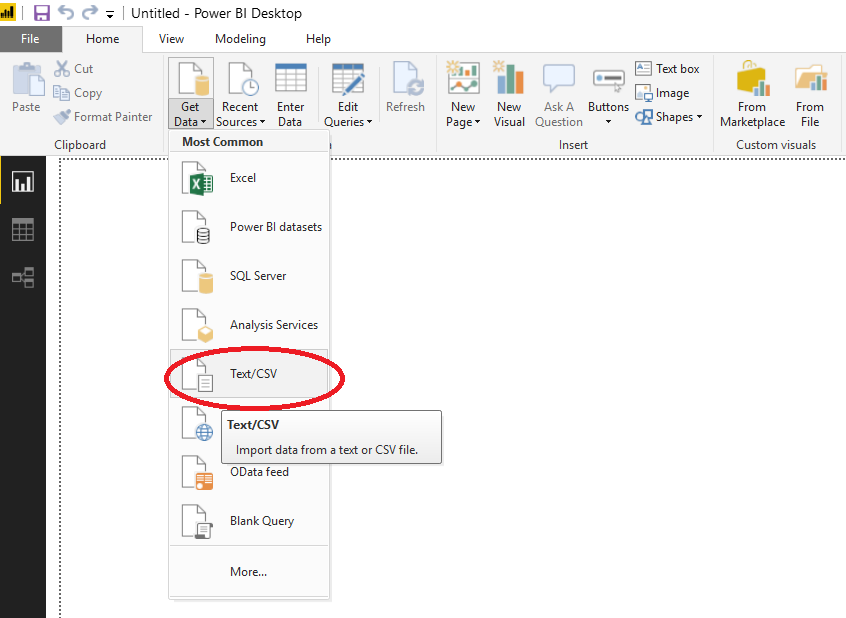
This data file is in the file is reduced to 10MB, this contains sample clickstream of visitors coming to the website. As you can see, that this data has a time-series component. So, I will be using to perform some data visualization and would also write a spark job on streaming click data

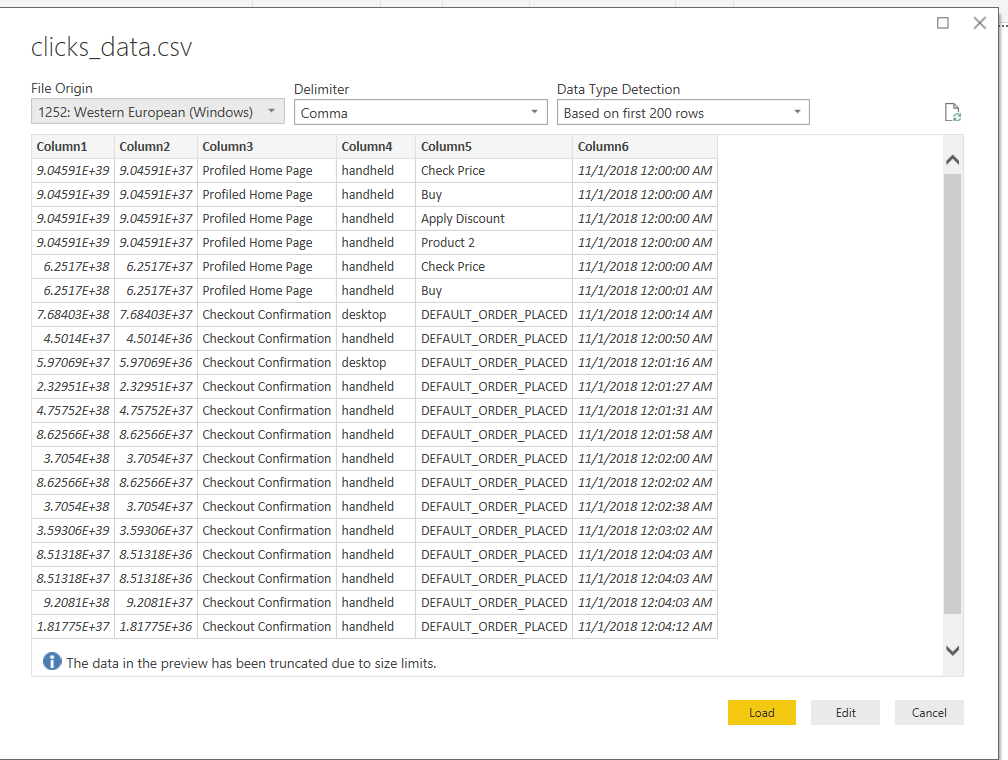
### **Data Visualization Exploratory Data Analysis**:

For this part, I have used Power BI dashboards – since power BI is drag and drop components there is not much coding – needed except for some ETL and data modifications.

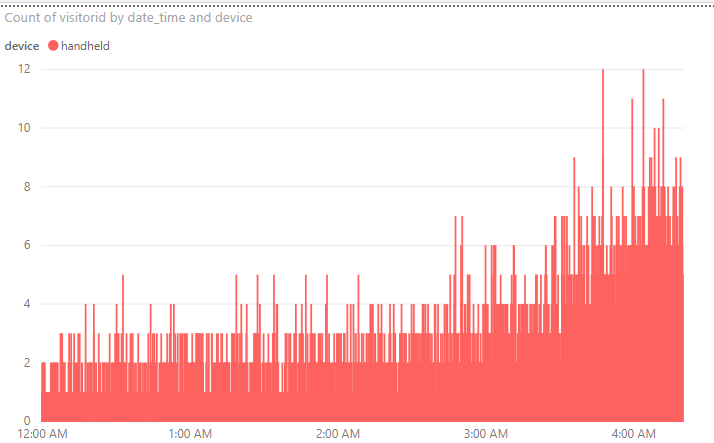


First, we have to get data – and then upload the clicks\_data.csv file to the Power BI environment



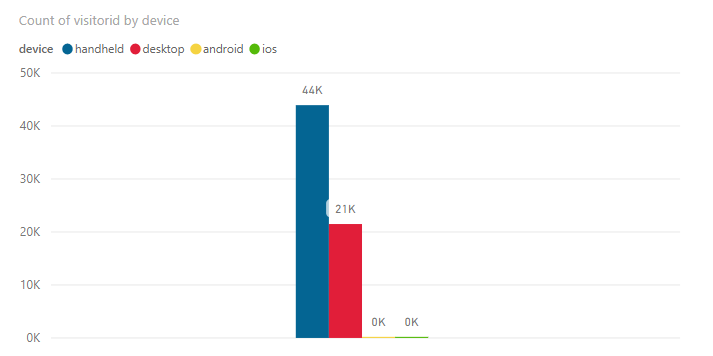


***(Dashboard attached with the submission)***



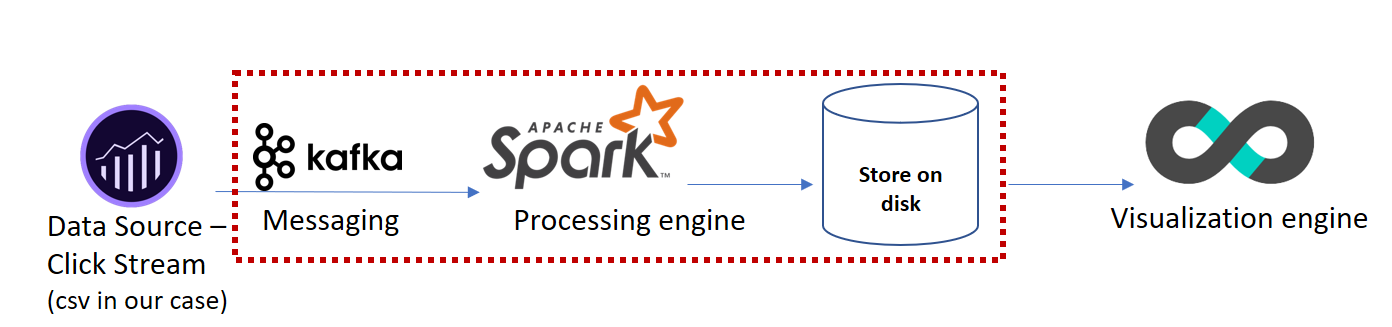
### 

### From the sample data, it looks like this website has maximum traffic during early morning hours on both Desktop and handheld platforms



### I also looked at their traffic by device type – it looks like handheld traffic is almost twice of desktop traffic – while there is still a lot of catching up to do for android and iOS applications

### **Code:**



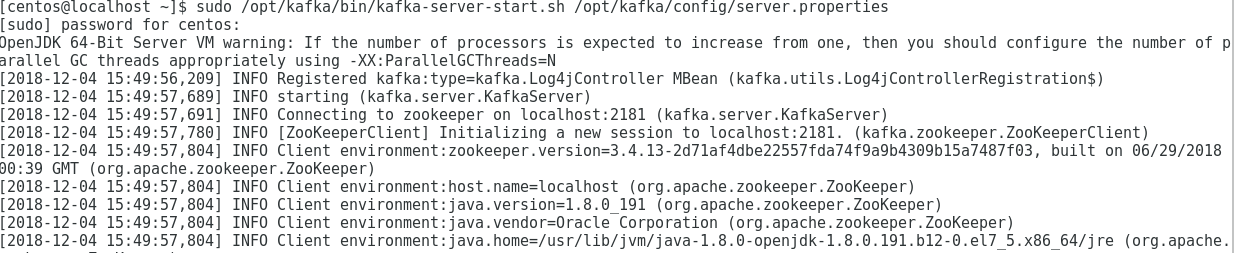
In this section of report, I will be covering the actual implementation of the project. It will cover creation of Kafka Topic, streaming data on Kafka topic, connecting Kafka Streams to Spark processing engine, and finally writing processed data on disk or database

To Start Kafka Server, first we need to start Zookeeper Services:

sudo/opt/kafka/bin/zookeeper-server-start.sh /opt/kafka/config/zookeeper.properties

Now I will start Kafka Server,

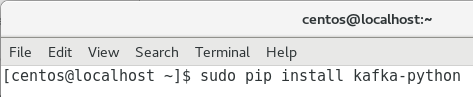
sudo /opt/kafka/bin/kafka-server-start.sh /opt/kafka/config/server.properties



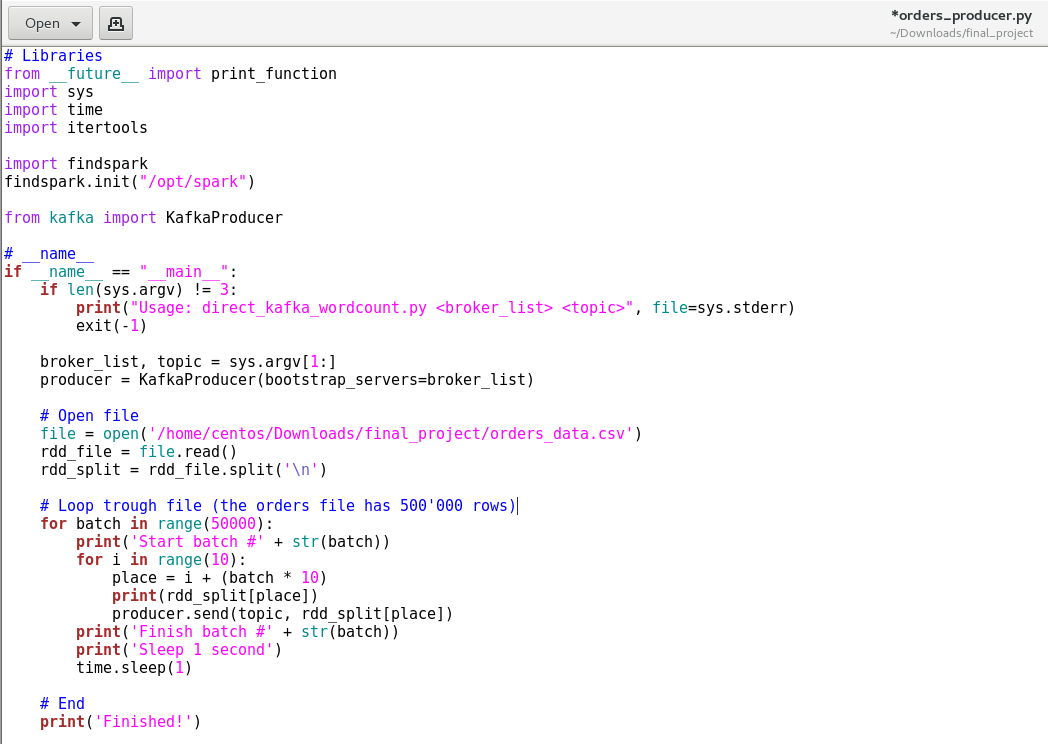
So once Kafka Server is running, I had to also download Kafka-python from pip so that we can run our producer and consumer codes

Downloading kafka-python using pip utility.

sudo pip install kafka-python



Code (orders\_producer.py): **(*Also attached with in source folder*)**



Over here, I am writing a producer code – which would send 10 data streams every second on a Kafka Topic as a batch. I wrote this code to create a simulation of a real-life clickstream data.

Now, we will write the consumer side code:

Code (orders\_update\_state\_by\_key.py): **(*Also attached with in source folder*)**

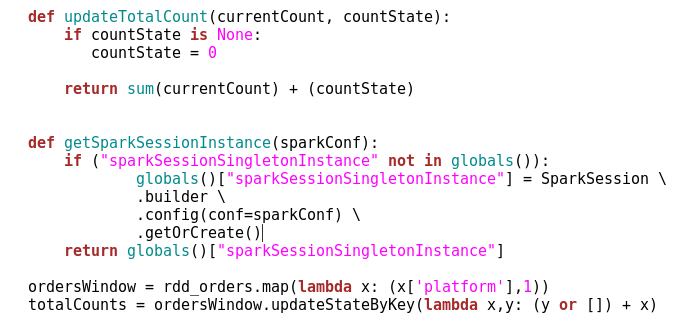


In this part of code, I am importing all the necessary libraries, writing a parser which would parse the orders data

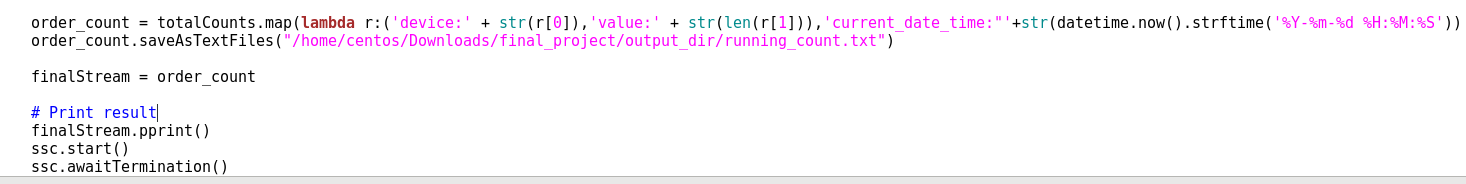
Next, I wrote an updateFunc where I am saving sum of previous result and adding to the new results

## 

In this part of code, I am creating Spark Context, Opening Kafka Streams and Parsing data streams



Over here, I am mapping stream data and mapping each order by device – after that I am adding previous count by keeping updatingStateByKey. This will give counts in form of 1’s



Now I am mapping data to device, value and I am also adding current\_date\_time – which is processing time of the row

Then I am saving files on disk by using saveAsTextFiles command.

Creating a topic first

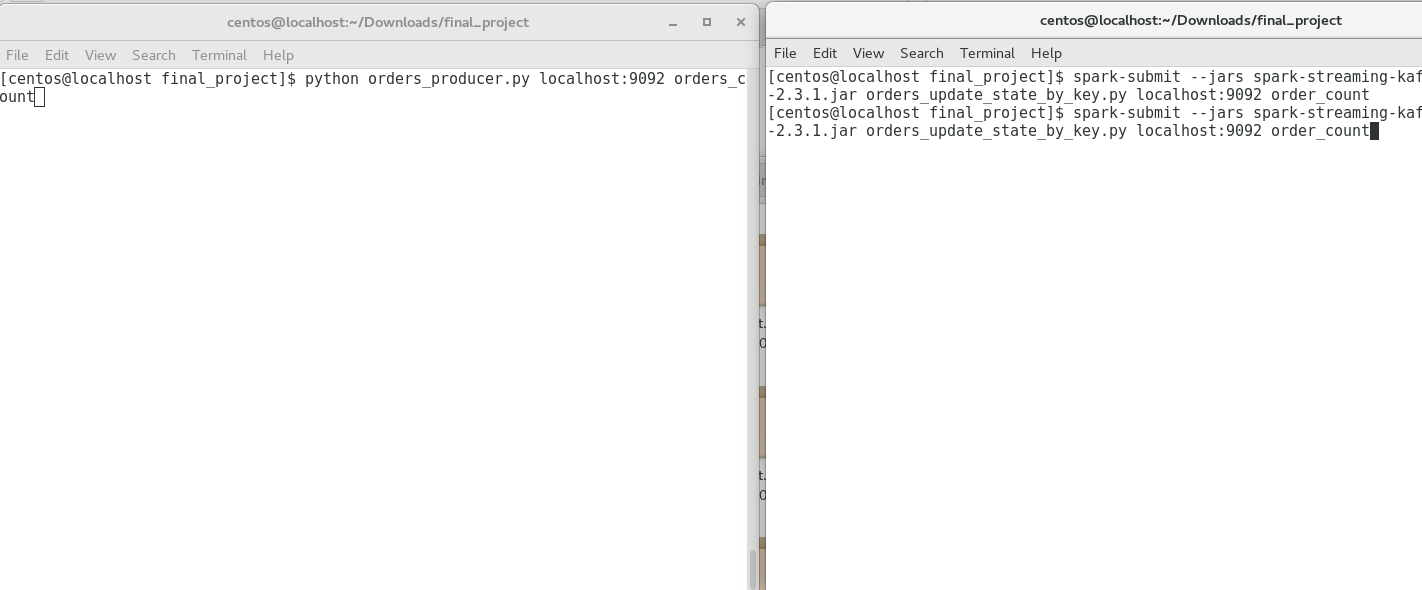
**sudo /opt/kafka/bin/kafka-topics.sh --create --zookeeper localhost:2181 \**

**--replication-factor 1 --partitions 1 --topic order\_count**

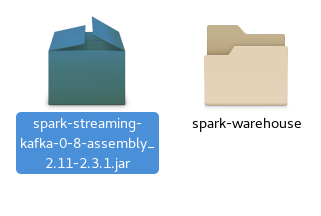
Now I will be running the producer and consumer code together to show the code running and the processing

**Producer code: python orders\_producer.py localhost:9092 order\_count**

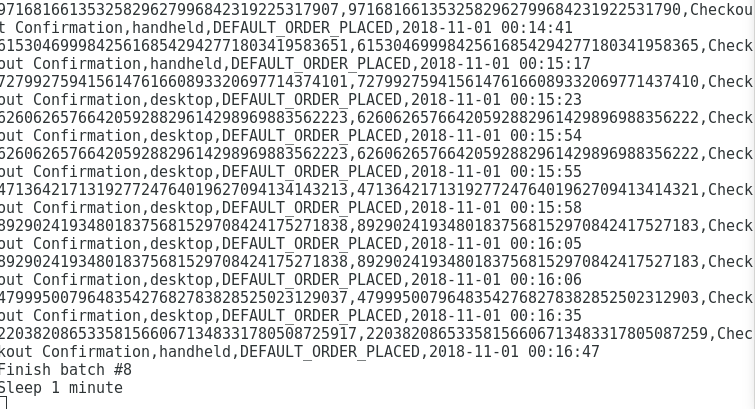
**Consumer code: spark-submit --jars spark-streaming-kafka-0-8-assembly\_2.11-2.3.1.jar orders\_update\_state\_by\_key.py localhost:9092 order\_count**



Before Running the code – we also must make sure that there is a right Kafka-Spark connector jar

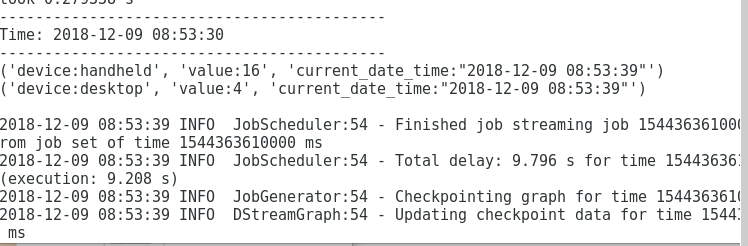


**Output on Producer Terminal**



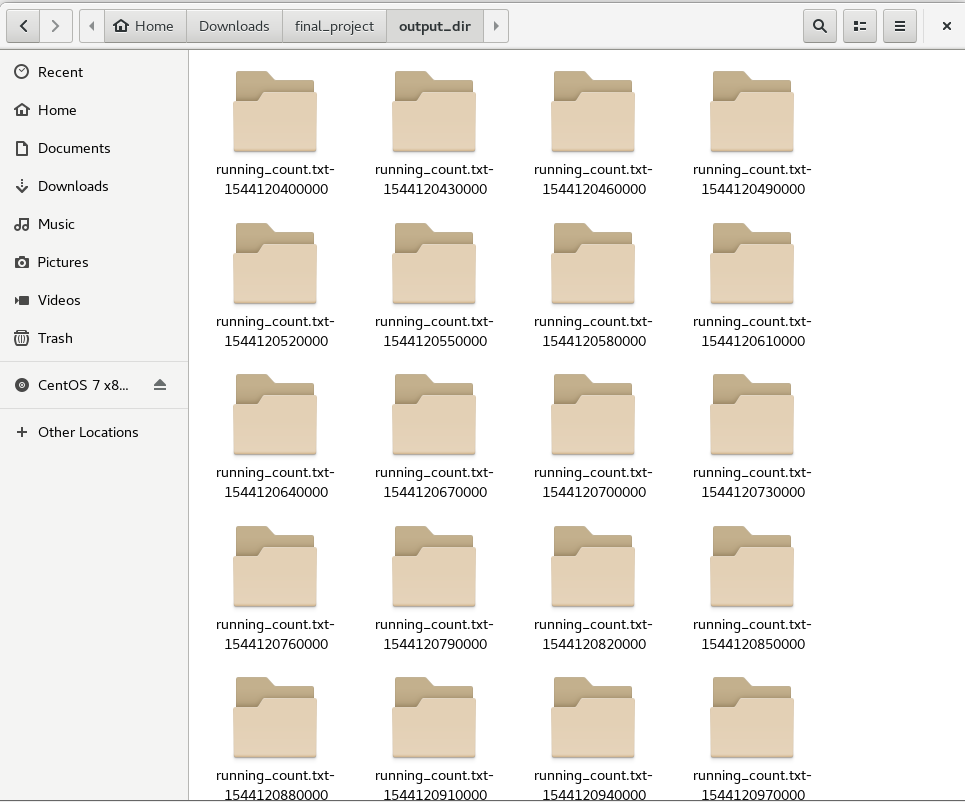
Producer is sending data on order\_count topic – which is all the order information of each visitor

**Output on Consumer Terminal**



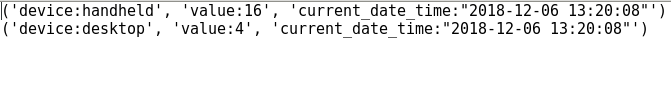
Consumer will start aggregating results in real-time and start writing this information on the disk. The challenge was to make sure – updateStatebyKey works on Dstreams – and I was successfully able to maintain state on counts in consumer code

Now as we see, that our consumer code wrote output files as part files, therefore for any meaningful visualization we need to write some code which could combine all these times in real-time and send these results to a real-time dashboard

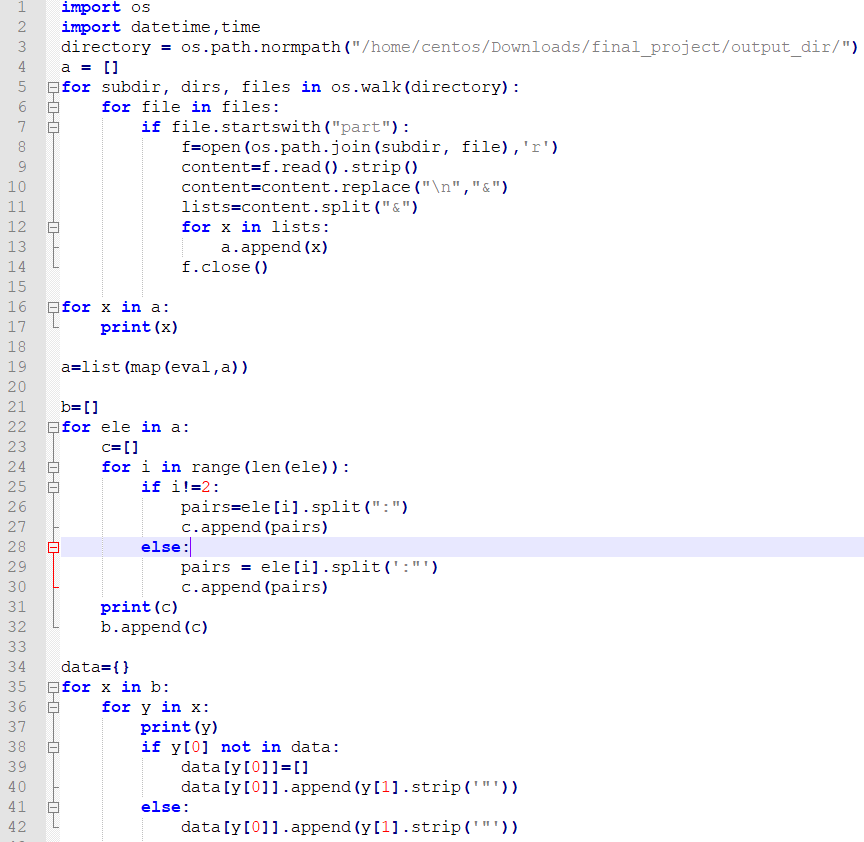


So, to solve this problem, I wrote a python code which would read all the part files from output directory

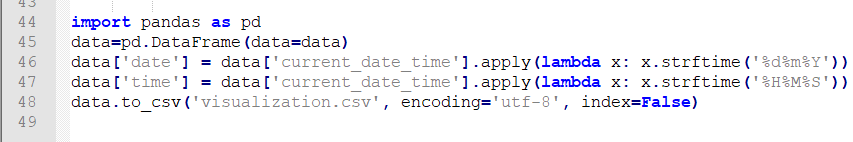
**This is sample output file**



Code (combining\_output\_files.py): **(*Also attached with in source folder*)**



In above code, I am reading files from output directory – where python is reading every part file inside the output directory after that I am doing a lot of ETL and data cleansing



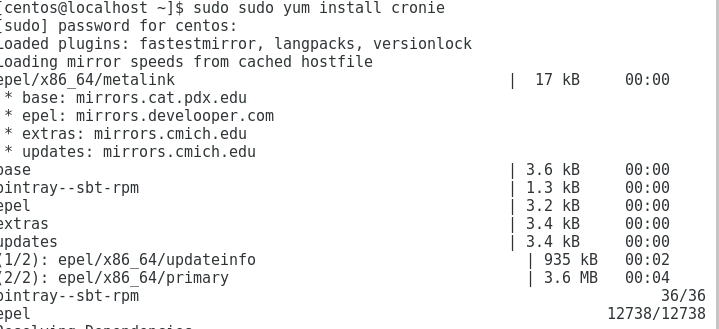
Importing files as pandas dataframe – and writing that dataframe as csv file named as visualization.csv

This is the output file which is generated (named as visualization.csv):



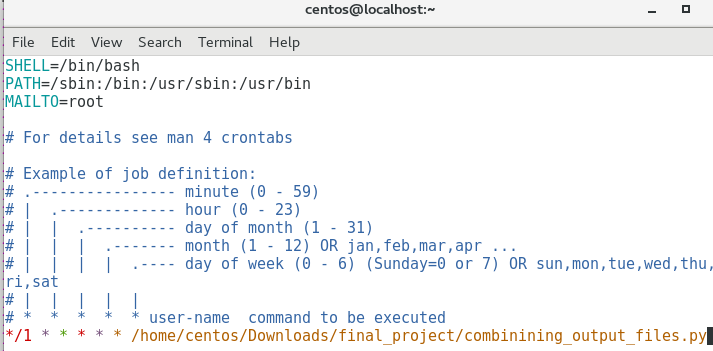
Now to keep updating this file periodically I put this file in cron job.

First, I must put this file under crontab



Checking the running of cron service

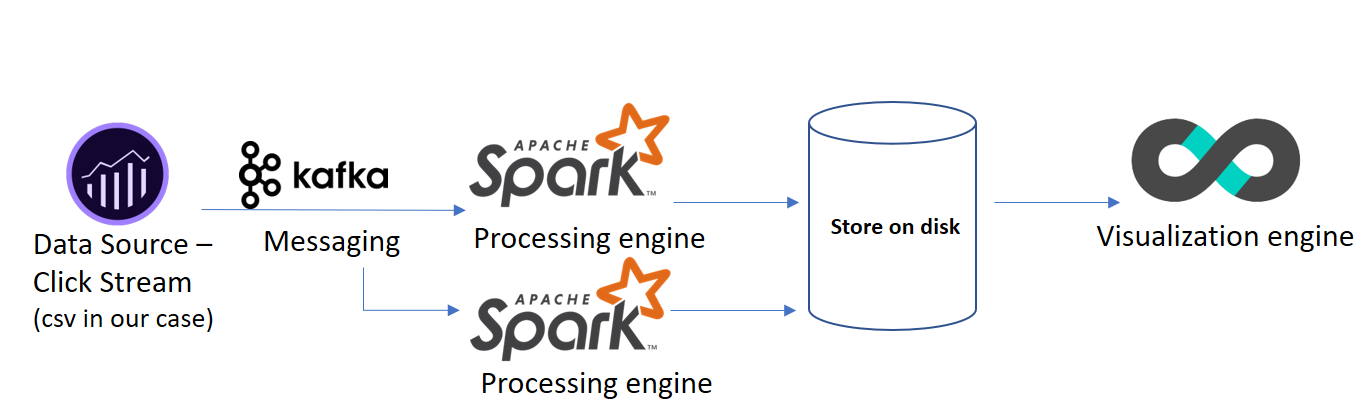
## 



Setting up this cron job – would run the script every minute and will update values in form of time-series chart shown as under:



So now I can run orders successfully, now I will do same for clickstream data and I will publish this data on clickstream\_topic, but I will be consuming this data stream by two consumers simultaneously (*showing power of spark-Kafka streaming)*



Creating a topic first

**sudo /opt/kafka/bin/kafka-topics.sh --create --zookeeper localhost:2181 \**

**--replication-factor 1 --partitions 1 --topic clickstream\_topic**

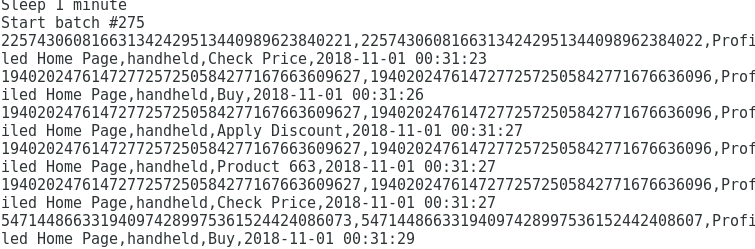
\*\* All codes are present in source folder \*\*

**Producer code: python clicks\_producer.py localhost:9092 clickstream\_topic**

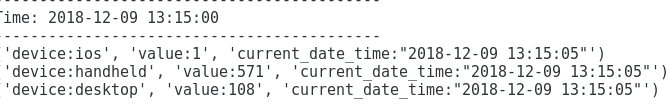
**Consumer code (1): spark-submit --jars spark-streaming-kafka-0-8-assembly\_2.11-2.3.1.jar orders\_update\_state\_by\_key.py localhost:9092 clickstream\_topic**

**Consumer code (2): spark-submit --jars spark-streaming-kafka-0-8-assembly\_2.11-2.3.1.jar clicks\_update\_state\_by\_key.py localhost:9092 clickstream\_topic**

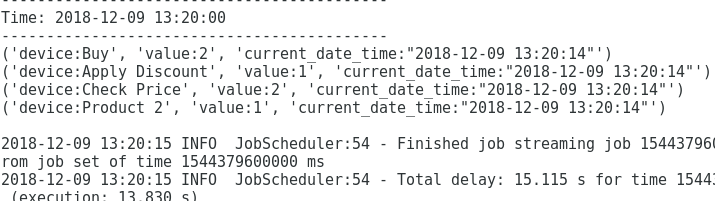
**Producer code output:**



**Consumer code (1) output:**



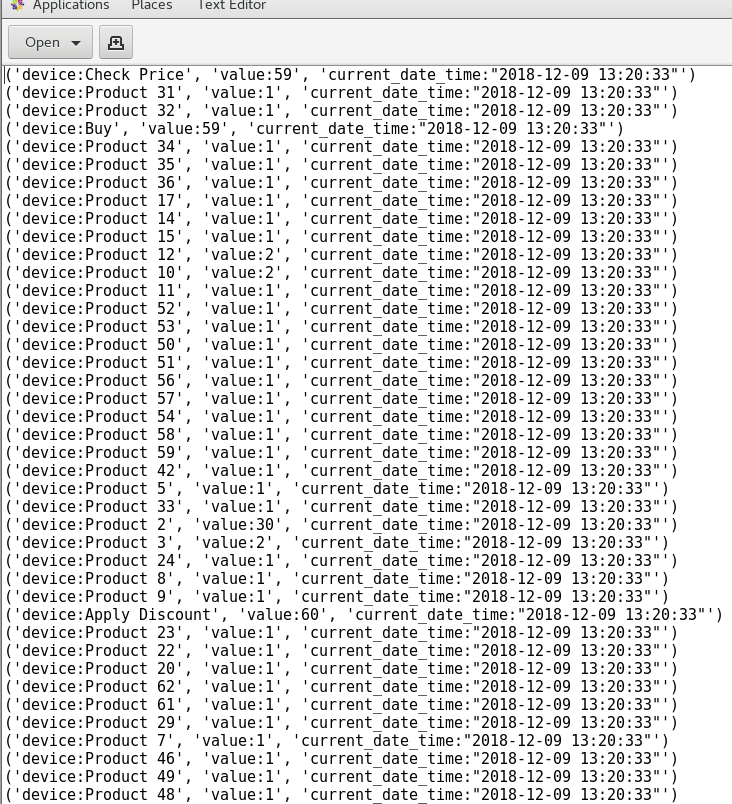
**Consumer code (2) output:**



In consumer code (2) we filtered all the streams on profiled homepage – as business wanted to know what the most clicked button on that page is so that they can optimize their page and increase Revenue optimization based on what button is clicked.

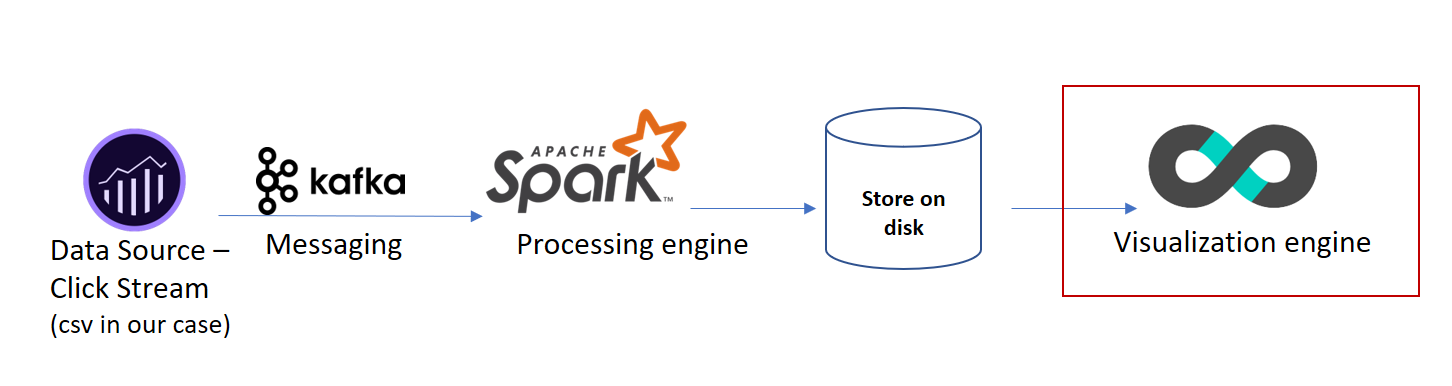


The below image shows output of output\_dir\_clicks, output\_dir should be same as previously shown results



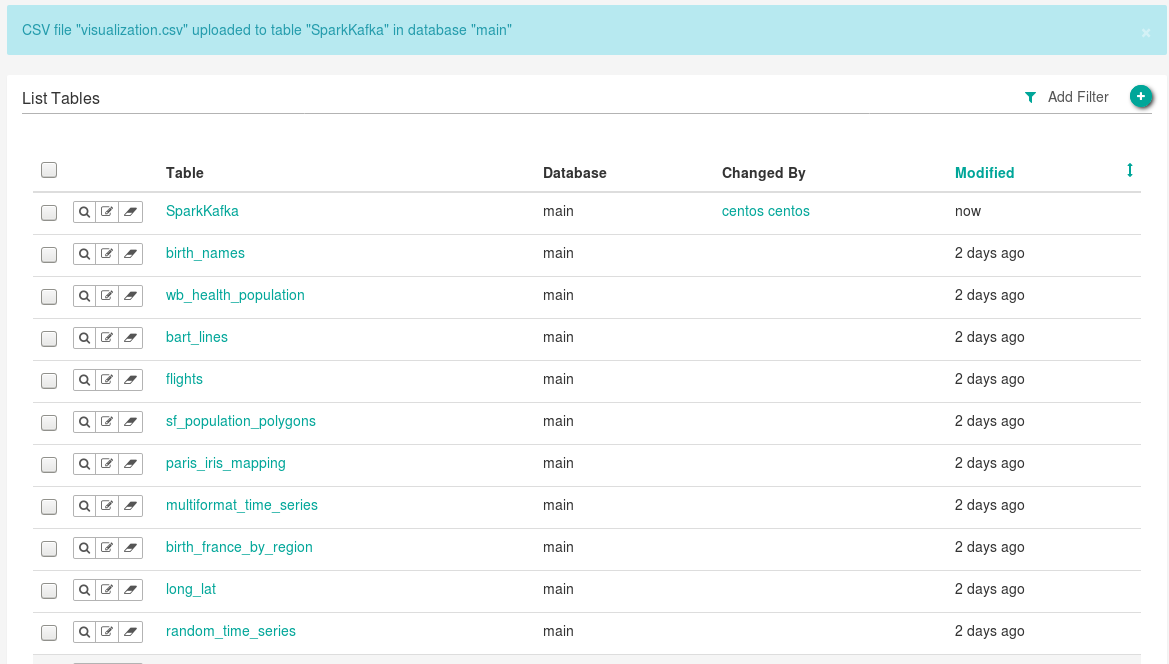
***Looks like people click more on ‘Check price’ on this page***

**Visualization**

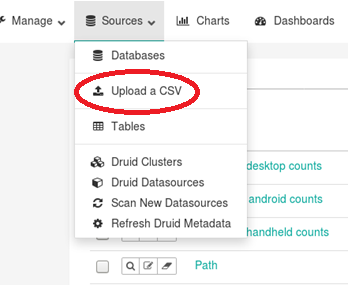


In this section of report, we will be taking the output of the results and use Apache Superset to visualize the results

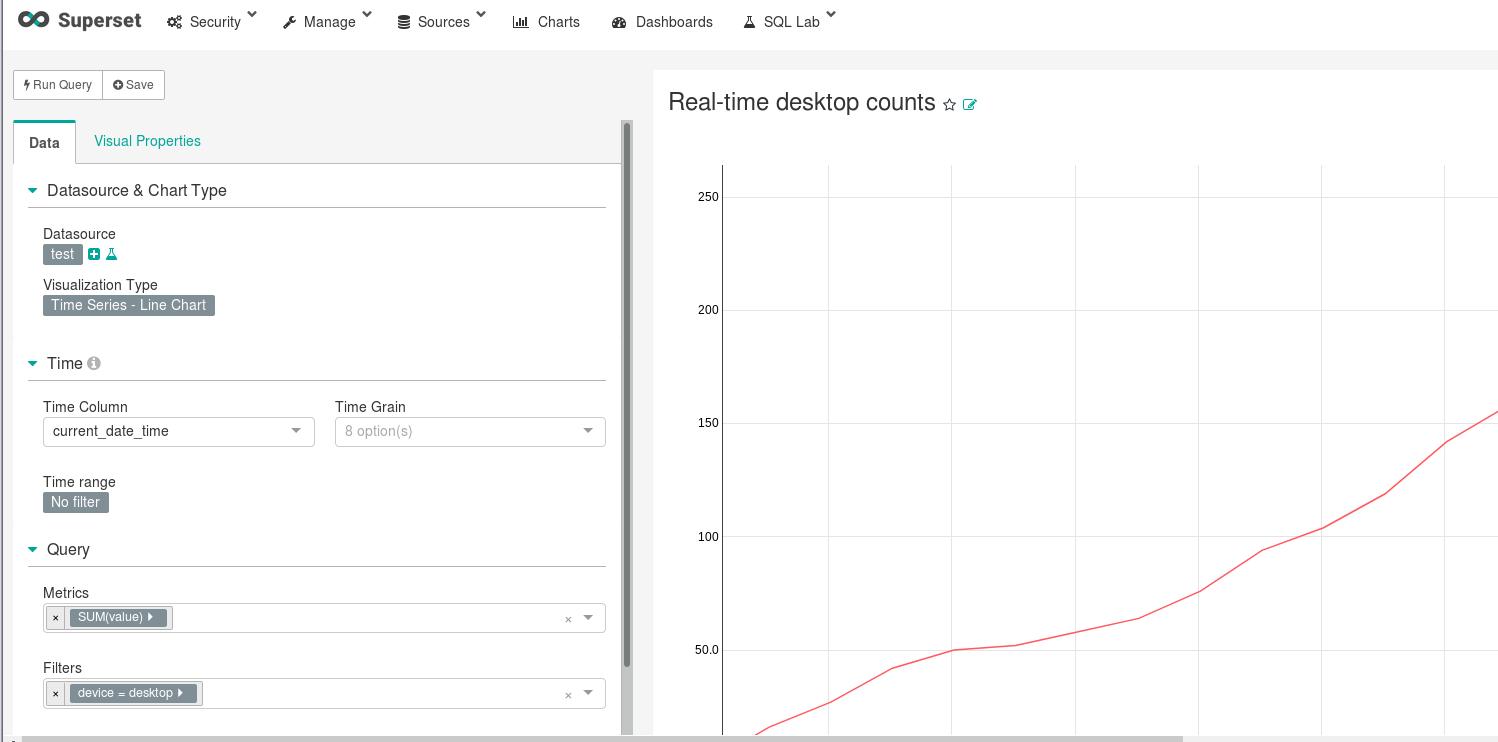
We started the server while installation now, starting dashboard at ***0.0.0.0:8088/*** in Mozilla Firefox browser

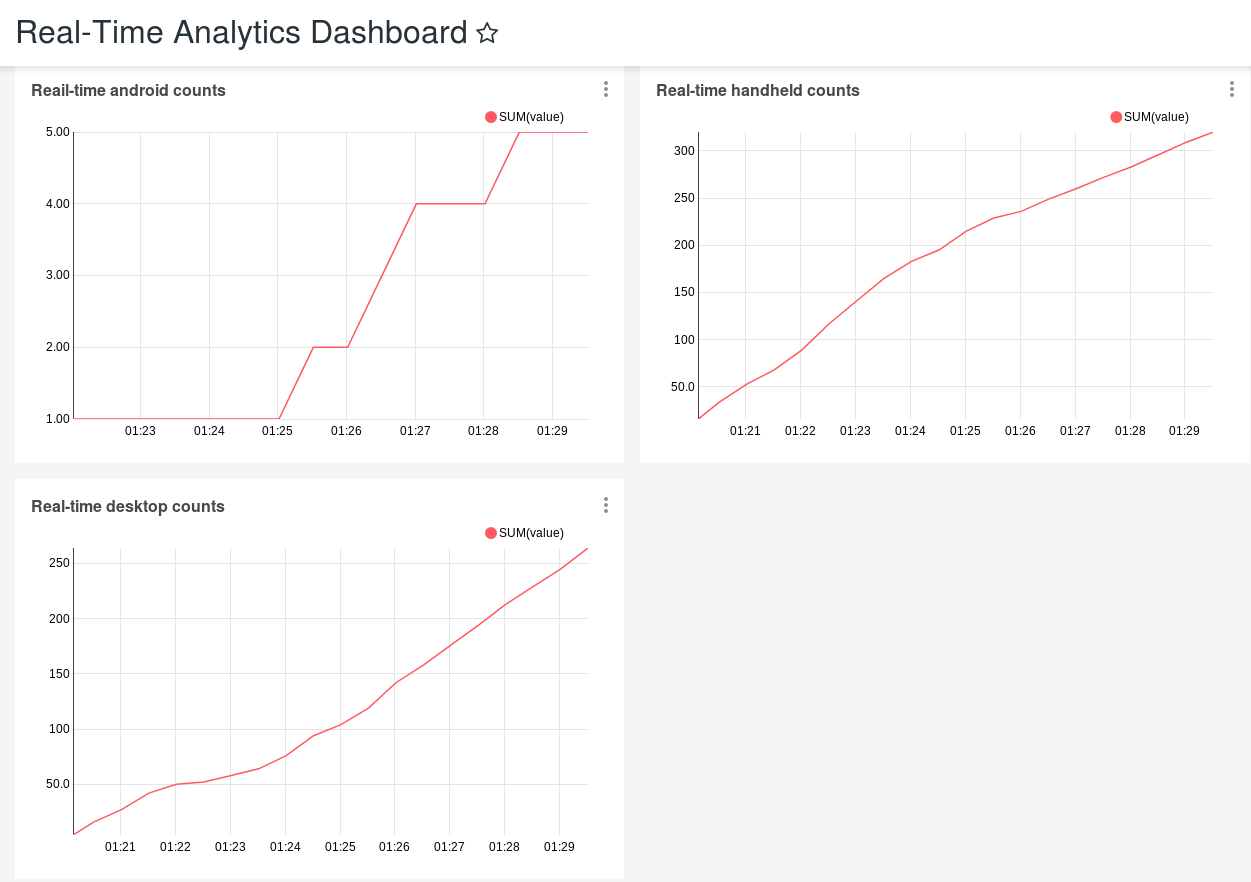


Uploading visualization.csv to the Apache superset database – Just follow next steps to upload the file



Creating a report, changing parameters and then I will be adding that into dashboard





So, this is setup for auto-refresh and will be updating numbers in real-time.

The above dashboard counts orders by each platform for time-series data – Looks like orders from handheld devices are more compared to desktop and android

As a future scope on this project would be using druid database instead of writing this on disk. and then auto-refreshing the dashboard

## **Conclusion**

Apache Kafka combined with Apache Spark processing gives us the computational power to do analysis in real-time

Using Kafka Streams an application can produce data streams at one end and can consume those data streams at multiple end clients.

Kafka processing makes stream processing a bit easier specially for an event-at-a-time processing. It is also useful for stateful processing including distributed joins and aggregations (what we used in our project).

The **downside** of Kafka Spark infrastructure is -- setting up of the infrastructure itself (specially in making frameworks to communicate with each other).

Another challenge can be working with streaming data **(Dstreams)** where one must be cognizant of the state of the stream. One other challenge be conversion of (Dstreams to RDD to Spark DF)

Nonetheless, once the infrastructure has been correctly setup: Kafka- Spark infrastructure can be useful for real-time applications. As a result, it has been widely adopted by the industry and companies like New York Times, Airbnb, Trivago, Netflix, etc. use Kafka and Spark for their analytics engines.

**References:**

* [*https://www.statista.com/statistics/379112/e-commerce-share-of-retail-sales-in-us/*](https://www.statista.com/statistics/379112/e-commerce-share-of-retail-sales-in-us/)
* [*https://towardsdatascience.com/introduction-to-druid-4bf285b92b5a*](https://towardsdatascience.com/introduction-to-druid-4bf285b92b5a)
* *Code-blocks from CSCI E-63 Big Data Analytics Fall 2018 Class*
* Databricks Spark Definitive Guide ([GitHub Link](https://github.com/databricks/Spark-The-Definitive-Guide/tree/master/data/retail-data/by-day))
* [*https://medium.com/@trK54Ylmz/real-time-dashboard-with-kafka-and-spark-streaming-53fd1f016249*](https://medium.com/@trK54Ylmz/real-time-dashboard-with-kafka-and-spark-streaming-53fd1f016249)

**You Tube URLs:**

2min preview presentation video: <https://youtu.be/EkGusepOISw>

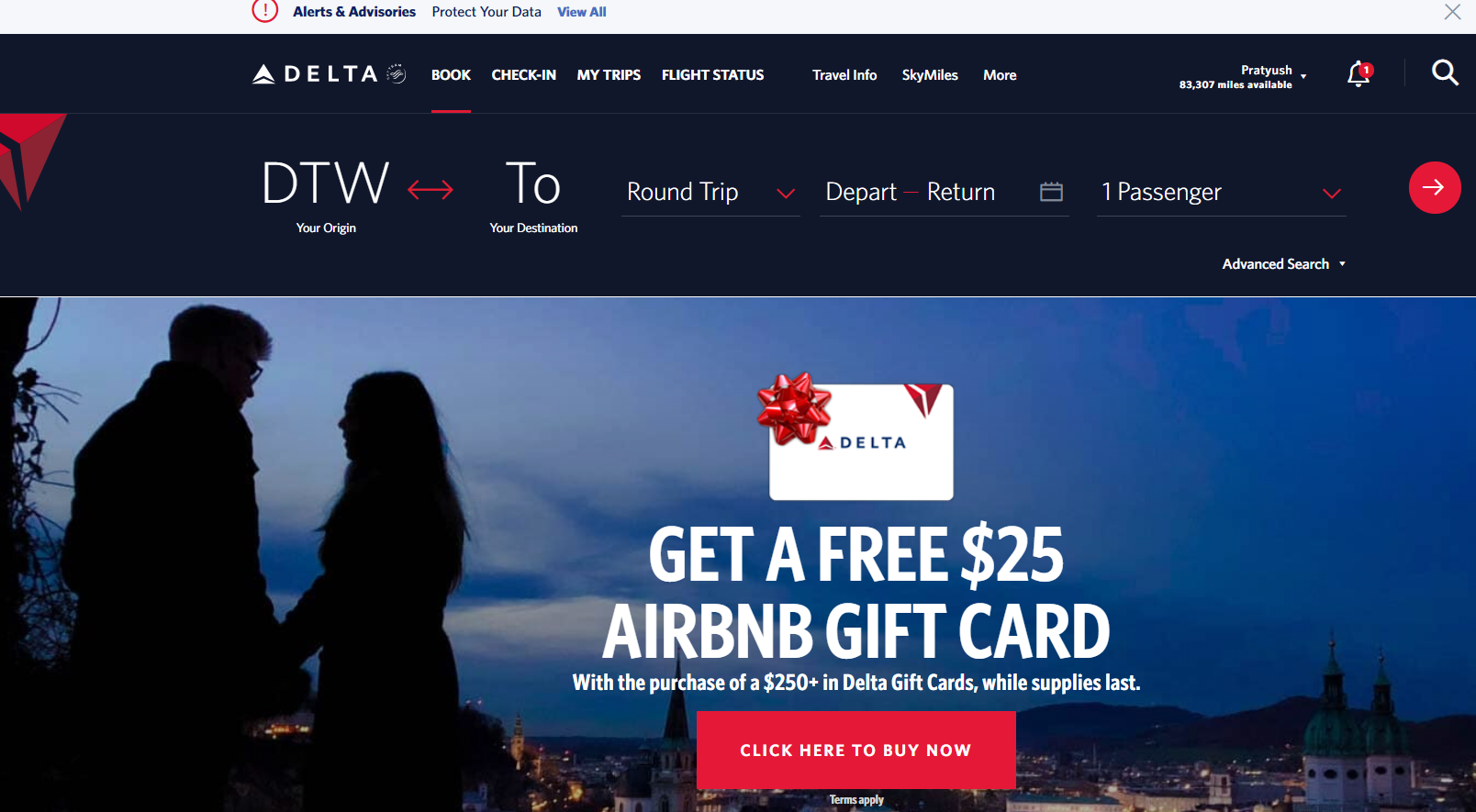
15min preview presentation video: <https://www.youtube.com/watch?v=WPPFJ1Dh2DI&t=29s>

7 min Demo video: <https://youtu.be/3HWcTSNimK0>

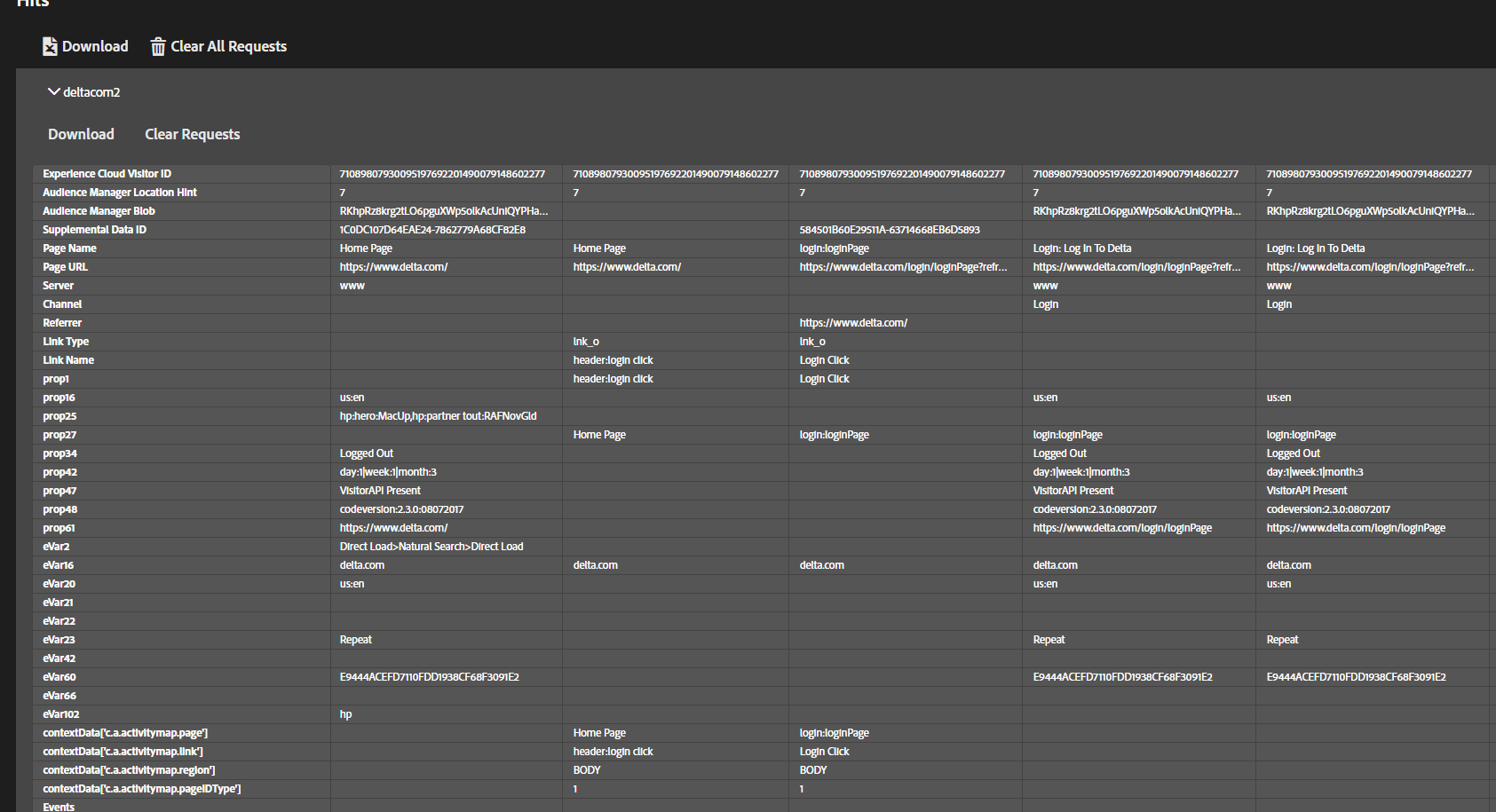
*(Note: According to video requirements – professor mentioned it is ok – to have 2 or 3-minute-long videos- therefore I added additional demo video)*

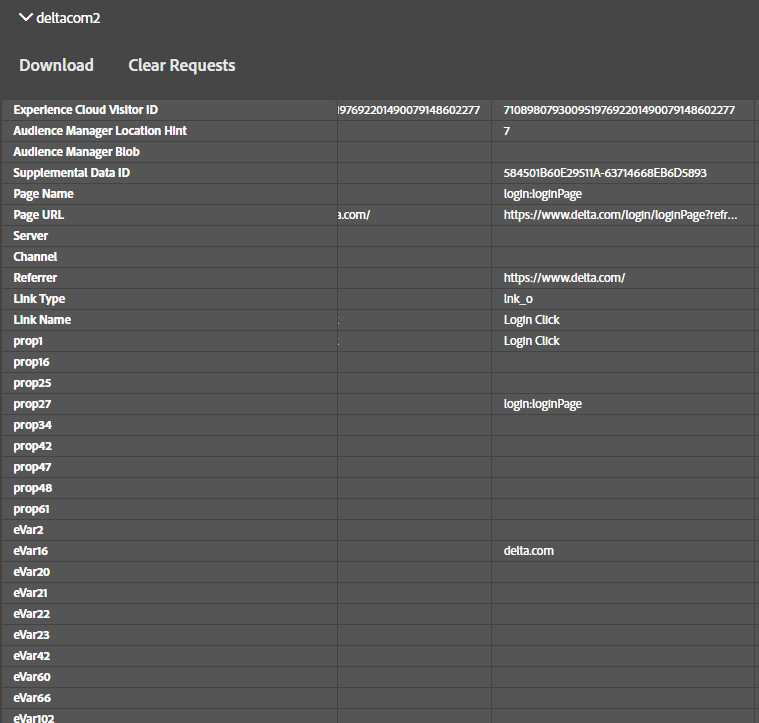
**Appendix Data generation:**

This is sample to show how data was generated, imagine you came to delta.com and now as soon as you come to the site – all the analytics tags start getting generated in back-end



This is screen shot of adobe analytics, at backend clickstream data gets generated





It’s publicly available clickstream data – you can implement tags on your site and do analytics as well.