Trending You Tube Videos Statistics

Veerendra Ramesh Kakarla, Mohith Kurakula

April 16, 2019

1. Introduction

YouTube allows users to upload videos, rate them, comment on them, add to playlist etc., also it maintains the list of trending videos based on user interactions (views, shares, likes, comments e.t.c). In general top performers in YouTube trending list are music videos, TV reality shows, Viral videos for which it is well known. It is also available with different versions based on different countries. It's interface suggests which version to be chosen based on IP address of the user. Individuals and large companies use this platform to increase their audience. So to help understand the actions of users based on different locations, also compare the views, likes of user from same location e.t.c, to perform analysis on the mentioned things we used data of top trending You Tube videos and this dataset is obtained using You Tube APIs where the datasets are obtained for different countries. Now that dataset is available from the APIs below we discuss about the dataset their corresponding attributes considered for the analysis, as well as the different approaches to be used to analyse and draw some insights between trending videos for different countries as well as comparisons of dataset from same country to know particular categories of videos liked or viewed by users.

2. Dataset

Dataset considered for this project includes months of data on daily trending YouTube videos. Data is considered for different countries like US, India, Britain e.t.c., with up to 100 listed trending videos per day. Each countries data is in separate csv file. Dataset contains below attributes:

video_id---Unique id for You Tube video.

trending_date---Trending data of that particular video.

title----Title of that particular video.

channel_title---Channel title from which the video is uploaded.

category_id---Category id to which the video belongs.

publish_time---Time when the video is uploaded to You Tube.

views---Number of views of that particular video.

likes--- Number of likes of that particular video.

dislikes--- Number of dislikes of that particular video.

comment_count---- Number of comments of that particular video.

description----Description of that particular video.

Above are the attributes of the dataset and these datasets are different for different countries and are stored in different files. But analysis between these datasets is done to get insights on trending You Tube videos based on different countries.

Here **category_id** field varies between country to country and the category_id is looked up from JSON file where category title is mapped to corresponding id.

Sample JSON is as below:

```
{
  "root":{
    "kind":"youtube#videoCategoryListResponse",
   "etag":"m2yskBQFythfE4irbTIeOgY/Xy1mB4_yLrHy ",
    "items":[
      {
        "kind":"youtube#videoCategory",
        "etag":"m2yskBQFythfE4irbTIeOgY/Xy1mB4_yLrHy ",
              "id":"1",-------------------CategoryID
```

```
            "snippet":{
              "channelid":"UCBR8-60-B28hp2BmDPdntcQ",
              "title":"Film & Animation",-------Category Title
              "assignable":true
      }
      }
      ]
}
}
```

The above dataset is obtained using YouTube API. Below is the source to the API: https://github.com/DataSnaek/Trending-YouTube-Scraper.

Data transformation to attributes like **trending_date**, **publish_time** which are in char are converted to date format for efficient analysis.

3.Framework Setup

The framework for the project as detailed uses Python-Flask as web server. We are using csv files in the project to read data. On the client side, we are using D3 for visualization. For asynchronous service calls from client to server (Flask) are made using AJAX http requests with JSON responses to accommodate the REST requests (service calls).

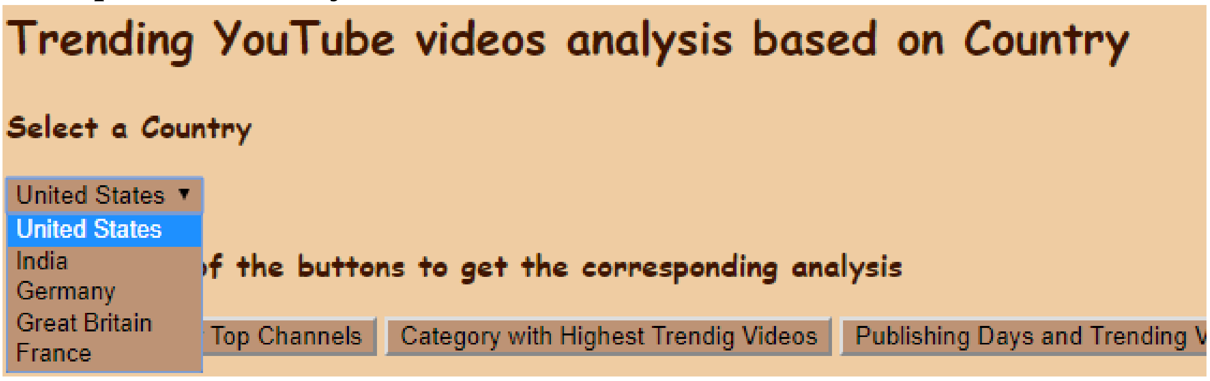
4.Proposal

4.1 Summary:

This project gives us some insights like how preferences of audience changes based on their locations, also we can get some information on category of videos that are top trending(in trending list for multiple days), videos that are most viewed, liked and disliked by the users where the datasets are considered for different countries, also some predictions which will be useful for channel owners to publish their videos so that they make it to trending list etc.

4.2 Insights & Visualizations:

In this project we used datasets of different countries to analyse the trending YouTube video's data and so user is provided with dropdown as shown below to get the corresponding analysis of that specific country selected.

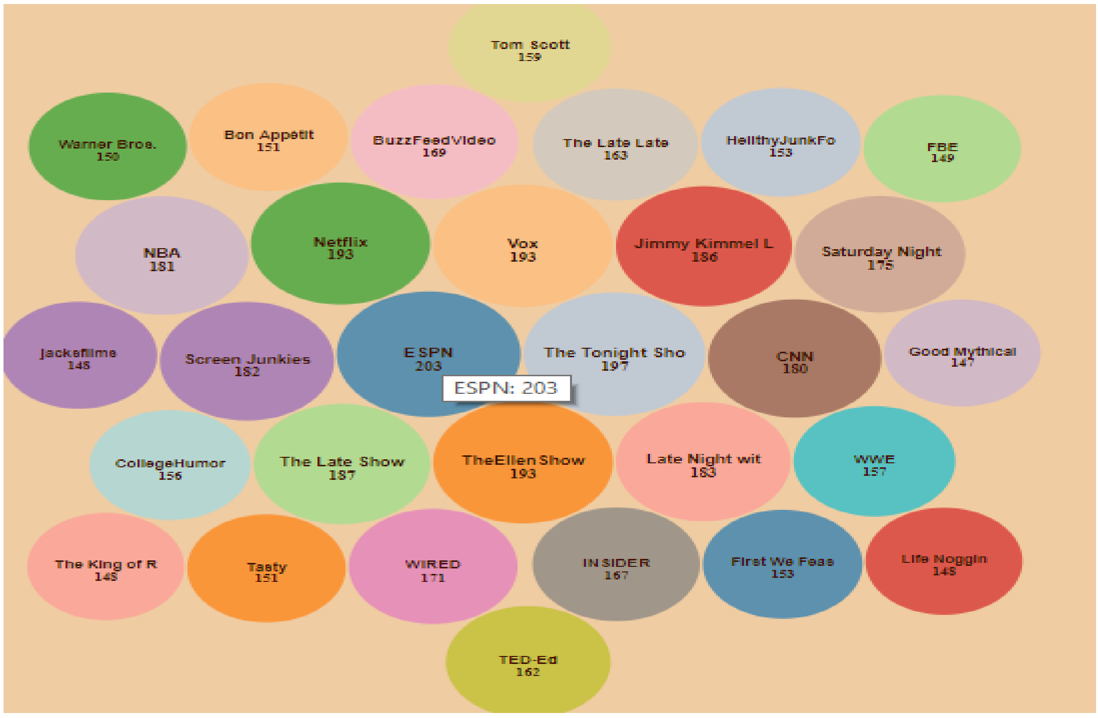


Trending YouTube videos analysis based on Country

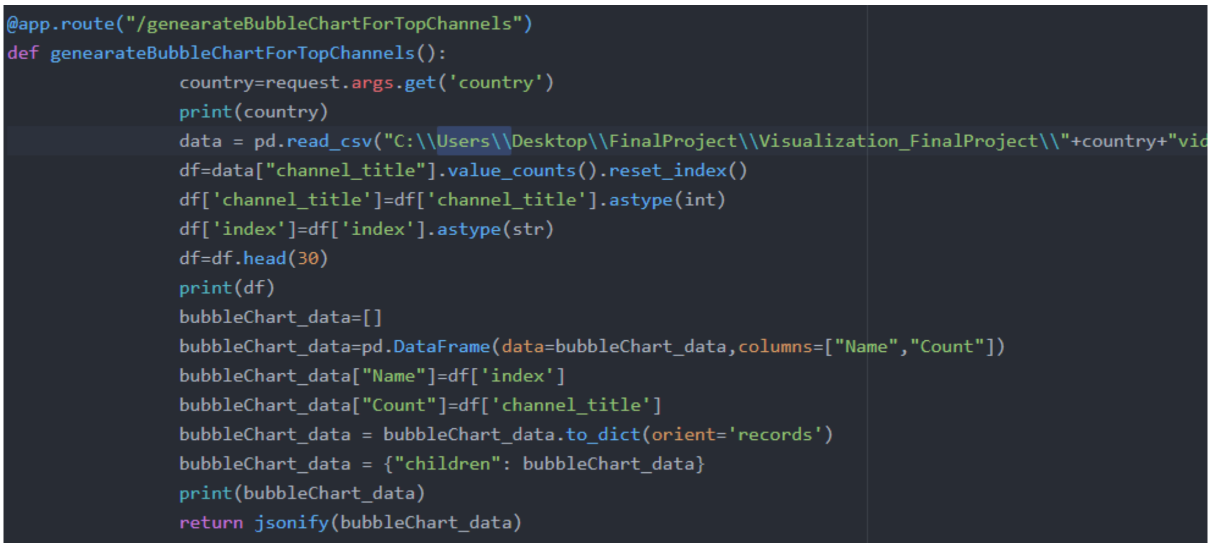
Select a Country

United States ▼
United States
India
Germany
Great Britain
France
f the buttons to get the corresponding analysis
Top Channels
Category with Highest Trendig Videos
Publishing Days and Trending V

Once the dropdown is available to the user, we first used Bubble chart to show the channels with highest videos in the trending list and the visualization of same is as below:

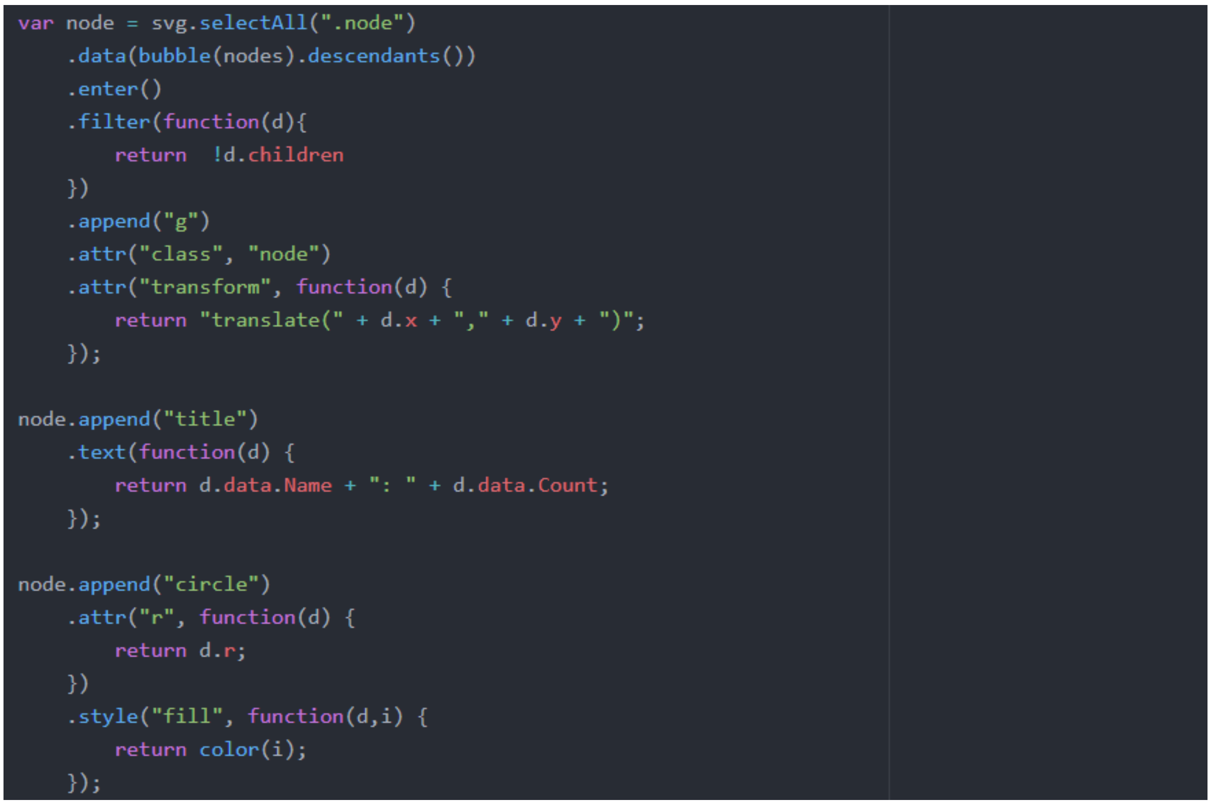


From above Bubble Chart we could conclude that ESPN channel has highest number of videos in the trending list when data set pf USA country is considered, the code snippets for the same is as below:



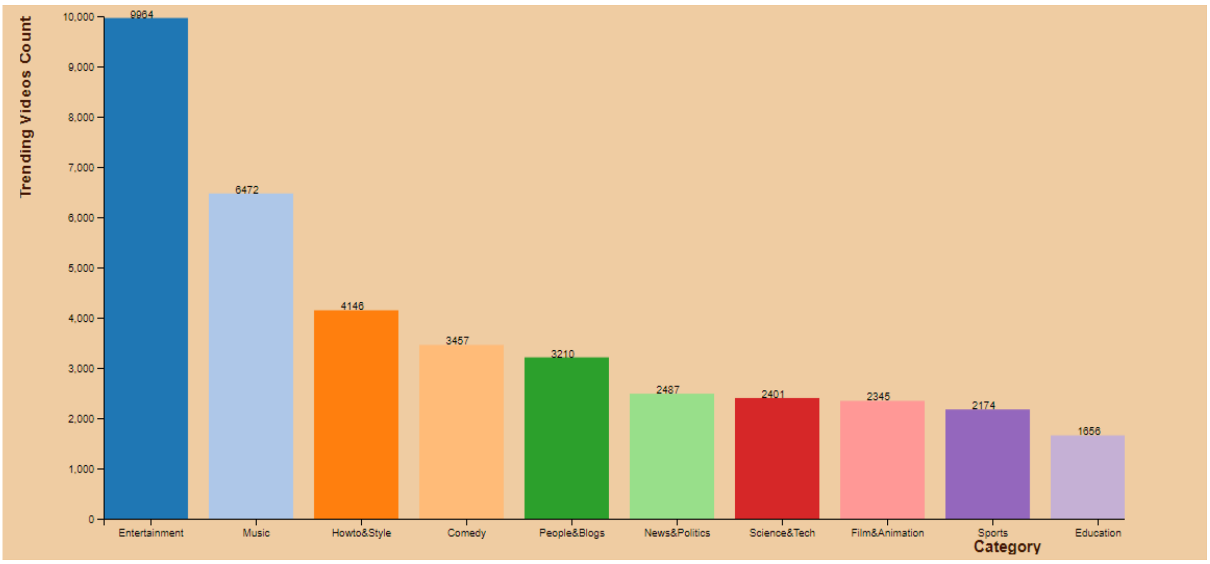
```python
@app.route("/genearateBubbleChartForTopChannels")
def genearateBubbleChartForTopChannels():
                country=request.args.get('country')
                print(country)
                data = pd.read_csv("C:\\Users\\Desktop\\FinalProject\\Visualization_FinalProject\\"+country+"vid
                df=data["channel_title"].value_counts().reset_index()
                df['channel_title']=df['channel_title'].astype(int)
                df['index']=df['index'].astype(str)
                df=df.head(30)
                print(df)
                bubbleChart_data=[]
                bubbleChart_data=pd.DataFrame(data=bubbleChart_data,columns=["Name","Count"])
                bubbleChart_data["Name"]=df['index']
                bubbleChart_data["Count"]=df['channel_title']
                bubbleChart_data = bubbleChart_data.to_dict(orient='records')
                bubbleChart_data = {"children": bubbleChart_data}
                print(bubbleChart_data)
                return jsonify(bubbleChart_data)
```

Above is the python code to get the highest count of trending videos based on channel title and the D3.js code to plot bubble chart is as below:

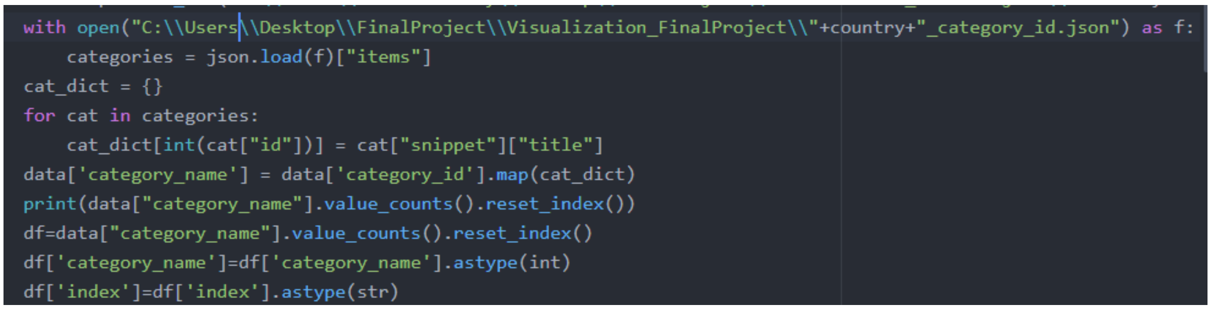


```javascript
var node = svg.selectAll(".node")
    .data(bubble(nodes).descendants())
    .enter()
    .filter(function(d){
        return  !d.children
    })
    .append("g")
    .attr("class", "node")
    .attr("transform", function(d) {
        return "translate(" + d.x + "," + d.y + ")";
    });

node.append("title")
    .text(function(d) {
        return d.data.Name + ": " + d.data.Count;
    });

node.append("circle")
    .attr("r", function(d) {
        return d.r;
    })
    .style("fill", function(d,i) {
        return color(i);
    });
```

Once user gets an insight about which types of channels has highest number of videos in Trending YouTube video’s list, a bar chart is used to get categories with highest number of Trending YouTube video’s and the visualization for the same is as below:

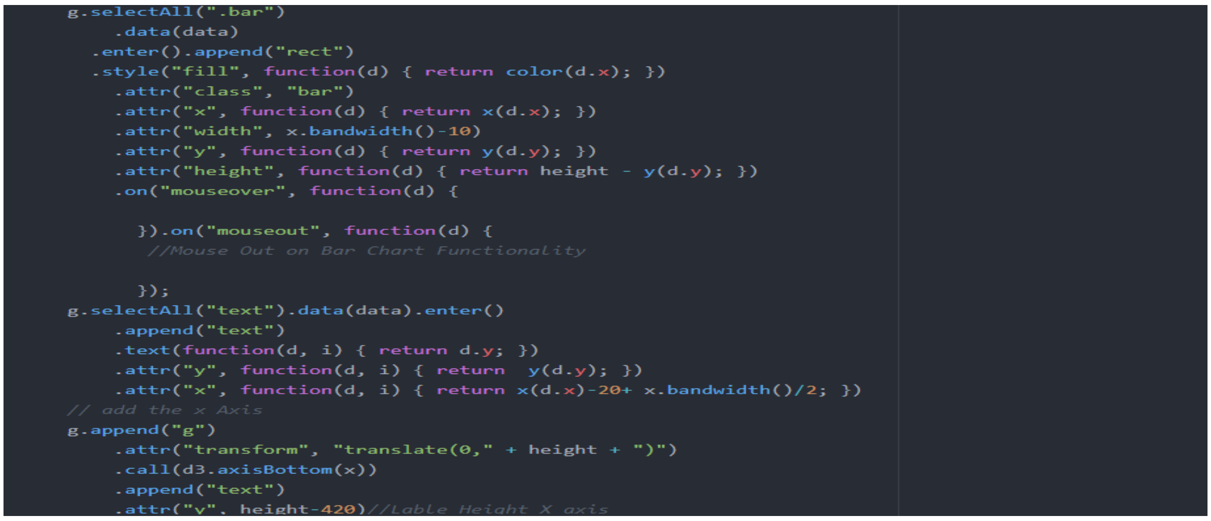


From above visualization we can conclude that in USA Entertainment and Music are the top 2 categories with highest number of videos in Trending List. Here the dataset has **category_id** as column so from that id using above mentioned JSON file in dataset description I look up for that corresponding category name based on **category_id** and the python code for the same is below:



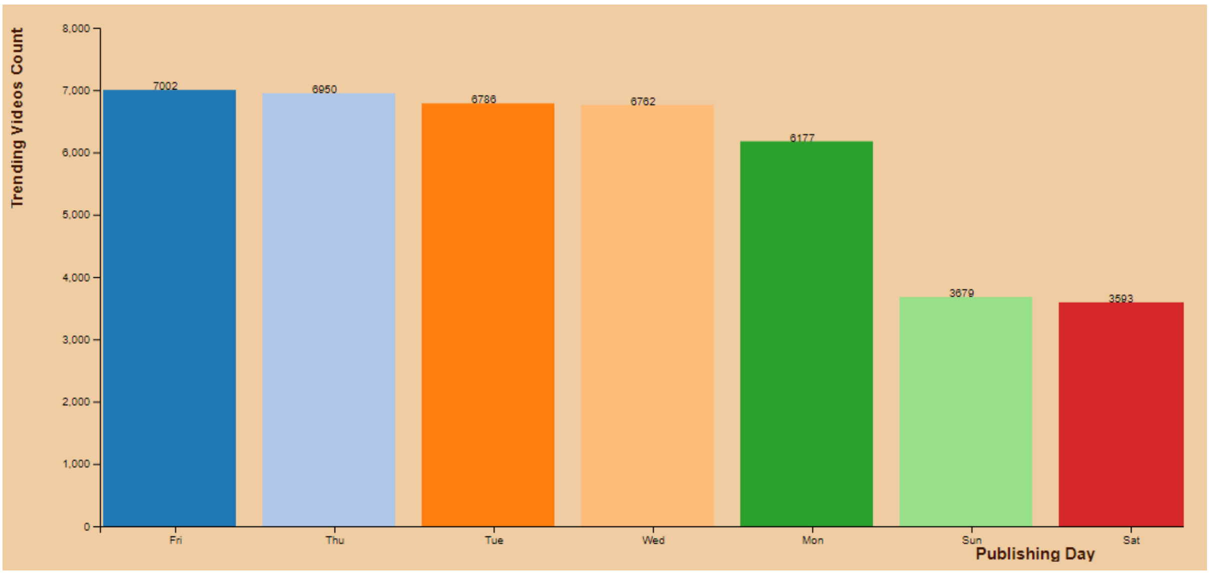
```
with open("C:\\Users\\Desktop\\FinalProject\\Visualization_FinalProject\\"+country+"_category_id.json") as f:
    categories = json.load(f)["items"]
cat_dict = {}
for cat in categories:
    cat_dict[int(cat["id"])] = cat["snippet"]["title"]
data['category_name'] = data['category_id'].map(cat_dict)
print(data["category_name"].value_counts().reset_index())
df=data["category_name"].value_counts().reset_index()
df['category_name']=df['category_name'].astype(int)
df['index']=df['index'].astype(str)
```

Once category name is available we used D3.js to plot bar chart and code for the same is as below:

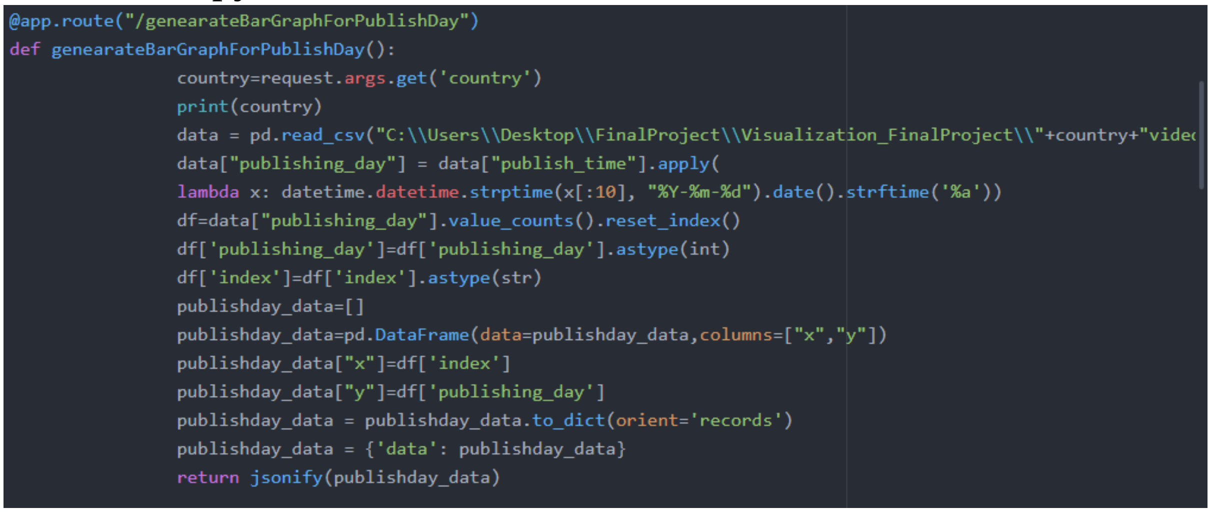


```
    g.selectAll(".bar")
        .data(data)
      .enter().append("rect")
      .style("fill", function(d) { return color(d.x); })
        .attr("class", "bar")
        .attr("x", function(d) { return x(d.x); })
        .attr("width", x.bandwidth()-10)
        .attr("y", function(d) { return y(d.y); })
        .attr("height", function(d) { return height - y(d.y); })
        .on("mouseover", function(d) {

          }).on("mouseout", function(d) {
           //Mouse Out on Bar Chart Functionality

          });
    g.selectAll("text").data(data).enter()
        .append("text")
        .text(function(d, i) { return d.y; })
        .attr("y", function(d, i) { return  y(d.y); })
        .attr("x", function(d, i) { return x(d.x)-20+ x.bandwidth()/2; })
    // add the x Axis
    g.append("g")
        .attr("transform", "translate(0," + height + ")")
        .call(d3.axisBottom(x))
        .append("text")
        .attr("y", height-420)//Lable Height X axis
```

Now that user knows which categories has highest number of videos in the trending list, another insight obtained using this dataset is particular publish days which has highest number of videos in the trending list and the visualization for the same is as below:

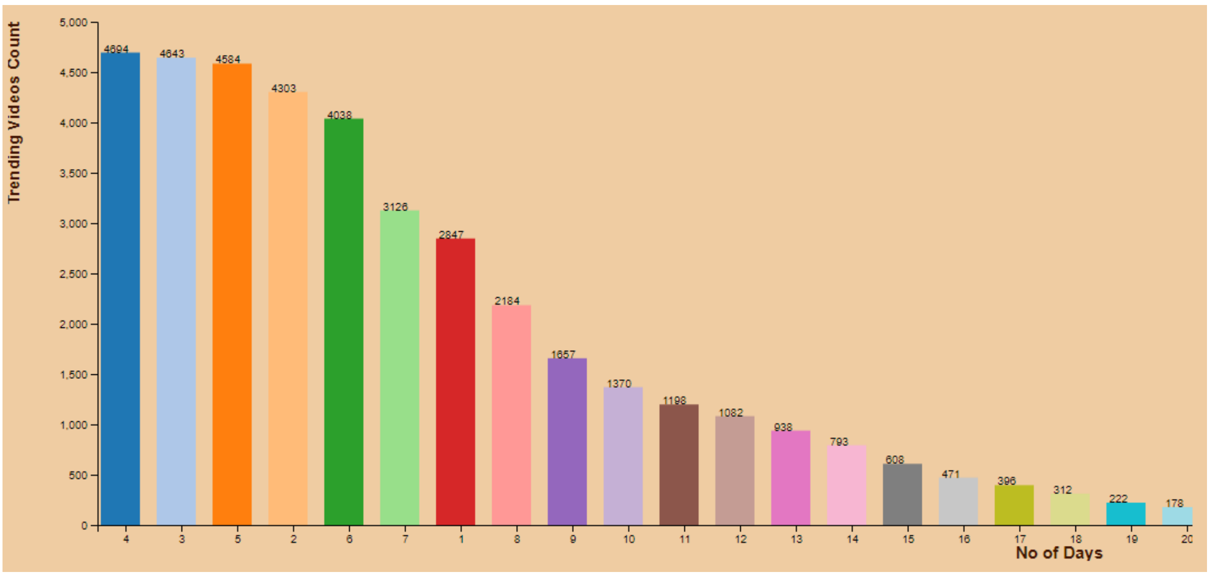


From above visualization we can conclude that in USA the videos that are published on Friday's are the highest in trending list, and the python code for the same is as below:

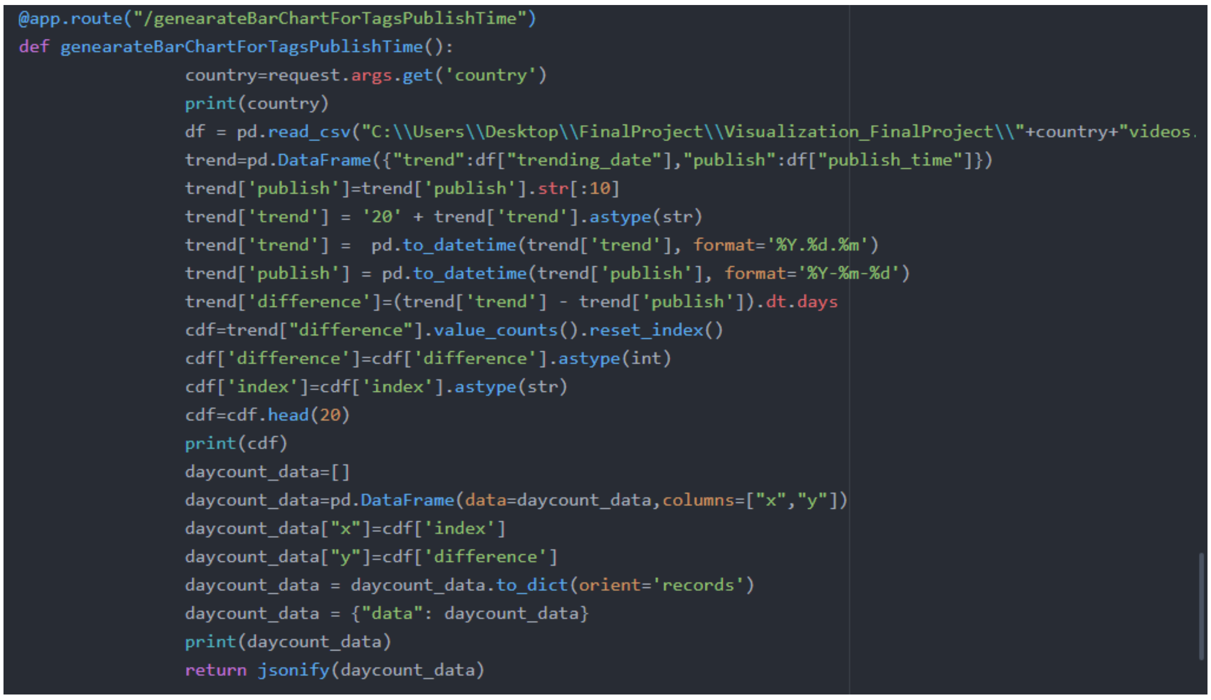


```
@app.route("/genearateBarGraphForPublishDay")
def genearateBarGraphForPublishDay():
            country=request.args.get('country')
            print(country)
            data = pd.read_csv("C:\\Users\\Desktop\\FinalProject\\Visualization_FinalProject\\"+country+"vide
            data["publishing_day"] = data["publish_time"].apply(
            lambda x: datetime.datetime.strptime(x[:10], "%Y-%m-%d").date().strftime('%a'))
            df=data["publishing_day"].value_counts().reset_index()
            df['publishing_day']=df['publishing_day'].astype(int)
            df['index']=df['index'].astype(str)
            publishday_data=[]
            publishday_data=pd.DataFrame(data=publishday_data,columns=["x","y"])
            publishday_data["x"]=df['index']
            publishday_data["y"]=df['publishing_day']
            publishday_data = publishday_data.to_dict(orient='records')
            publishday_data = {'data': publishday_data}
            return jsonify(publishday_data)
```

For bar chart in D3.js the code mentioned is reused to obtain the visualization. There is one more interesting insight obtained from the available dataset, which is the days difference between publish data and trending data and the visualization for the same is as below:

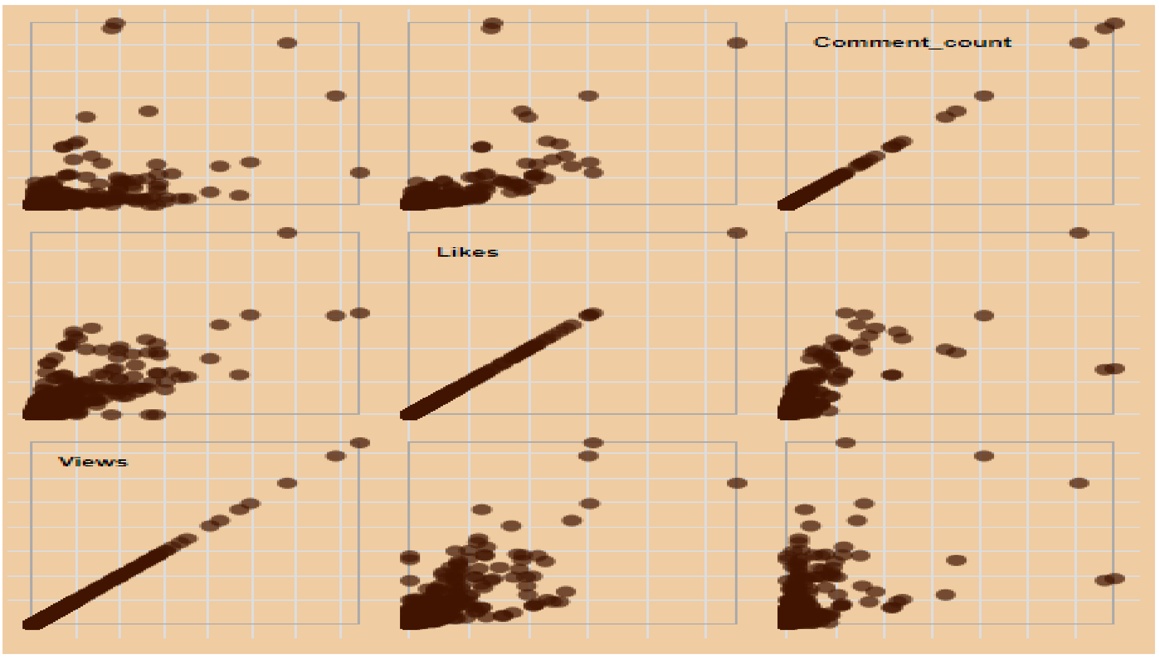


From above visualization we can conclude that the videos with days difference between publish date and trending date as 4 are more in number in trending list, so that means that it takes time for the video to enter into trending list and its very rare the video will be in trending list as soon as it is uploaded when USA dataset is considered and theses insights varied for each country for example same insight in case of India is like that the videos with days difference between publish date and trending date as 1 are more in number in trending list, and the python code for the same is as below:

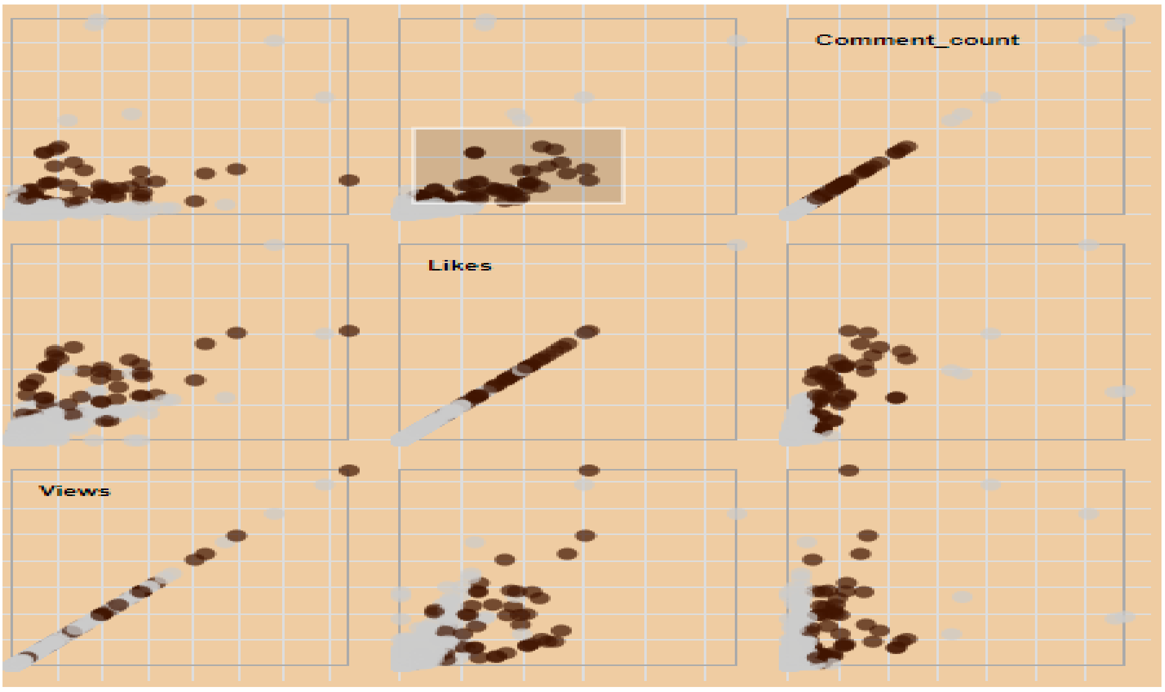


```
@app.route("/genearateBarChartForTagsPublishTime")
def genearateBarChartForTagsPublishTime():
                country=request.args.get('country')
                print(country)
                df = pd.read_csv("C:\\Users\\Desktop\\FinalProject\\Visualization_FinalProject\\"+country+"videos.
                trend=pd.DataFrame({"trend":df["trending_date"],"publish":df["publish_time"]})
                trend['publish']=trend['publish'].str[:10]
                trend['trend'] = '20' + trend['trend'].astype(str)
                trend['trend'] =  pd.to_datetime(trend['trend'], format='%Y.%d.%m')
                trend['publish'] = pd.to_datetime(trend['publish'], format='%Y-%m-%d')
                trend['difference']=(trend['trend'] - trend['publish']).dt.days
                cdf=trend["difference"].value_counts().reset_index()
                cdf['difference']=cdf['difference'].astype(int)
                cdf['index']=cdf['index'].astype(str)
                cdf=cdf.head(20)
                print(cdf)
                daycount_data=[]
                daycount_data=pd.DataFrame(data=daycount_data,columns=["x","y"])
                daycount_data["x"]=cdf['index']
                daycount_data["y"]=cdf['difference']
                daycount_data = daycount_data.to_dict(orient='records')
                daycount_data = {"data": daycount_data}
                print(daycount_data)
                return jsonify(daycount_data)
```

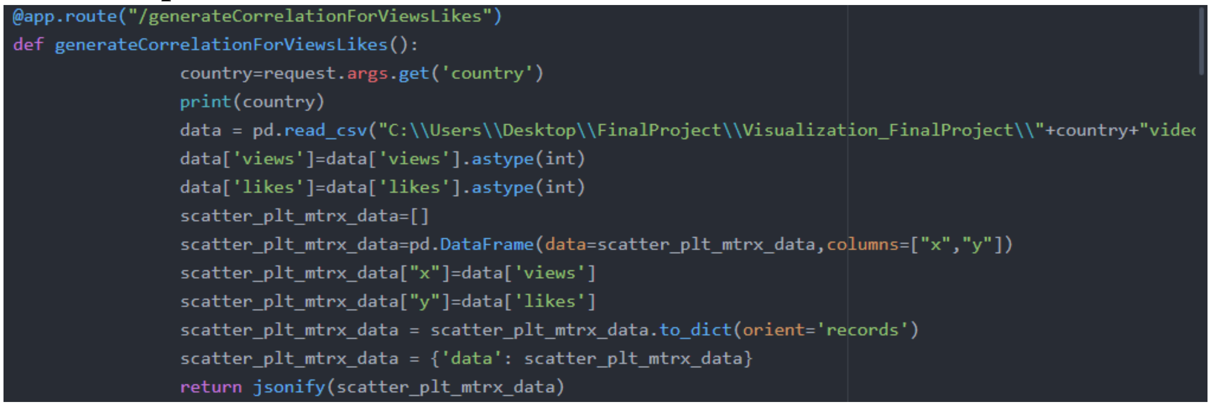
For bar chart in D3.js the code mentioned is reused to obtain the visualization. Scatter plot matrix is used to get the correlation between parameters like views, likes, comments_count and the visualization for the same is as below:



Here Brushing is used to give an understanding to the user how the points of views, likes and comments_count are scattered and the visualization for the same is as below:

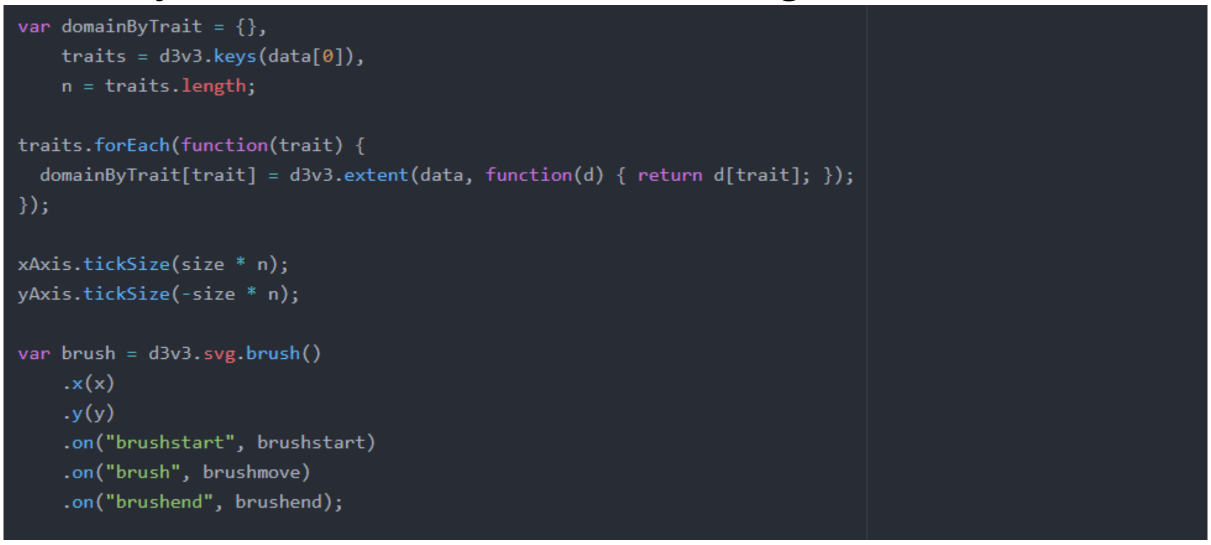


From above visualizations we can conclude that there is positive correlation between views and likes etc., Python code for the above implementation is as below:

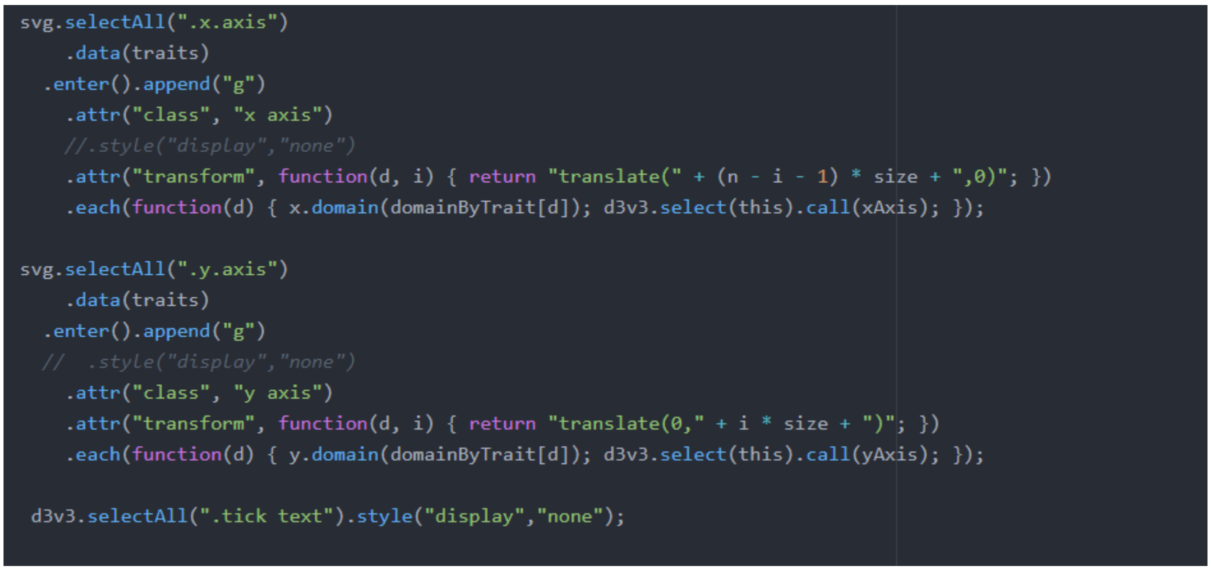


```
@app.route("/generateCorrelationForViewsLikes")
def generateCorrelationForViewsLikes():
                country=request.args.get('country')
                print(country)
                data = pd.read_csv("C:\\Users\\Desktop\\FinalProject\\Visualization_FinalProject\\"+country+"vide
                data['views']=data['views'].astype(int)
                data['likes']=data['likes'].astype(int)
                scatter_plt_mtrx_data=[]
                scatter_plt_mtrx_data=pd.DataFrame(data=scatter_plt_mtrx_data,columns=["x","y"])
                scatter_plt_mtrx_data["x"]=data['views']
                scatter_plt_mtrx_data["y"]=data['likes']
                scatter_plt_mtrx_data = scatter_plt_mtrx_data.to_dict(orient='records')
                scatter_plt_mtrx_data = {'data': scatter_plt_mtrx_data}
                return jsonify(scatter_plt_mtrx_data)
```

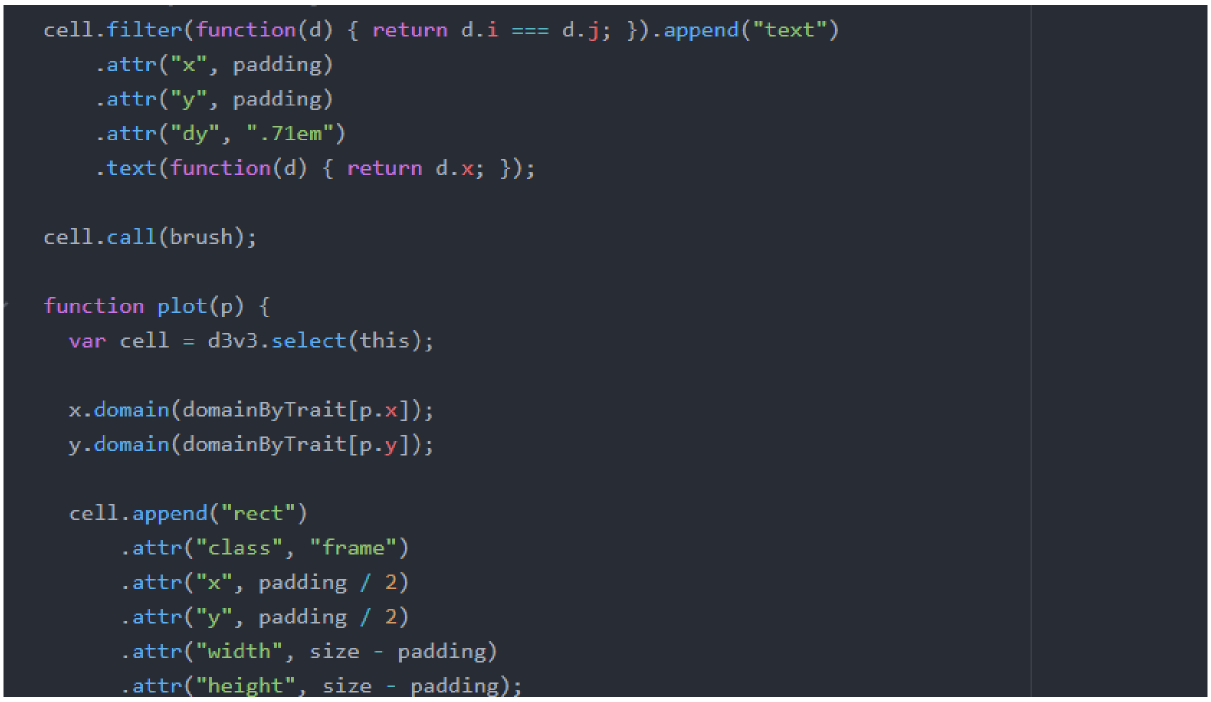
Also D3.js code for Scatter Plot with Brushing is as below:



```
var domainByTrait = {},
    traits = d3v3.keys(data[0]),
    n = traits.length;

traits.forEach(function(trait) {
  domainByTrait[trait] = d3v3.extent(data, function(d) { return d[trait]; });
});

xAxis.tickSize(size * n);
yAxis.tickSize(-size * n);

var brush = d3v3.svg.brush()
    .x(x)
    .y(y)
    .on("brushstart", brushstart)
    .on("brush", brushmove)
    .on("brushend", brushend);
```

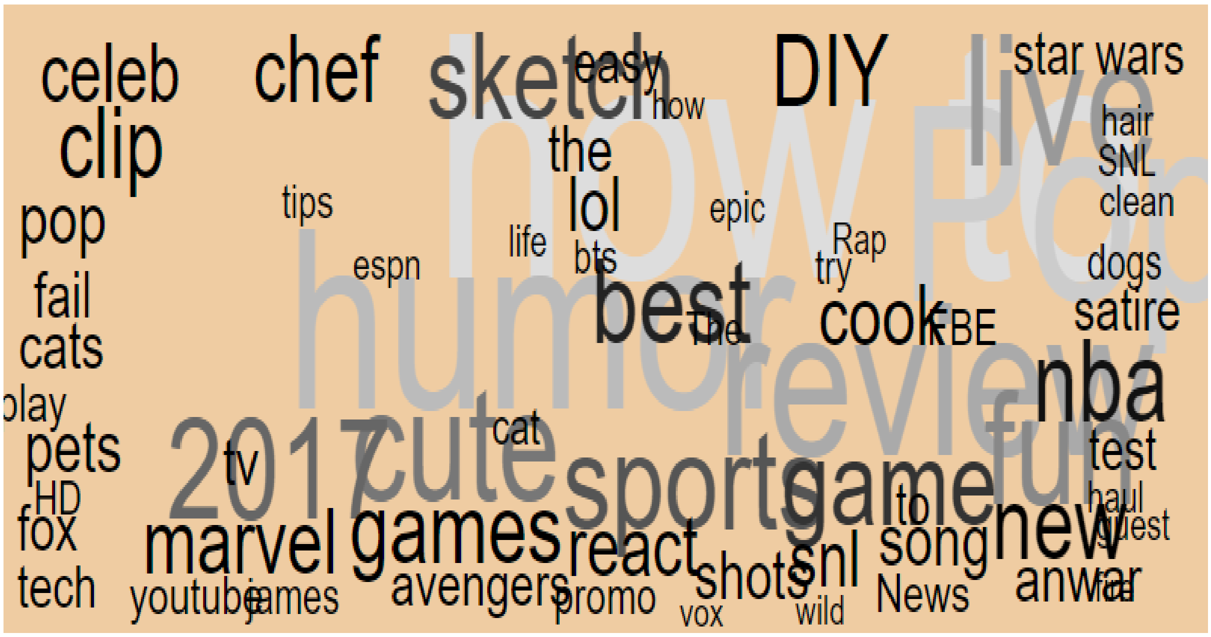


```
svg.selectAll(".x.axis")
    .data(traits)
  .enter().append("g")
    .attr("class", "x axis")
    //.style("display","none")
    .attr("transform", function(d, i) { return "translate(" + (n - i - 1) * size + ",0)"; })
    .each(function(d) { x.domain(domainByTrait[d]); d3v3.select(this).call(xAxis); });

svg.selectAll(".y.axis")
    .data(traits)
  .enter().append("g")
  //  .style("display","none")
    .attr("class", "y axis")
    .attr("transform", function(d, i) { return "translate(0," + i * size + ")"; })
    .each(function(d) { y.domain(domainByTrait[d]); d3v3.select(this).call(yAxis); });

 d3v3.selectAll(".tick text").style("display","none");
```

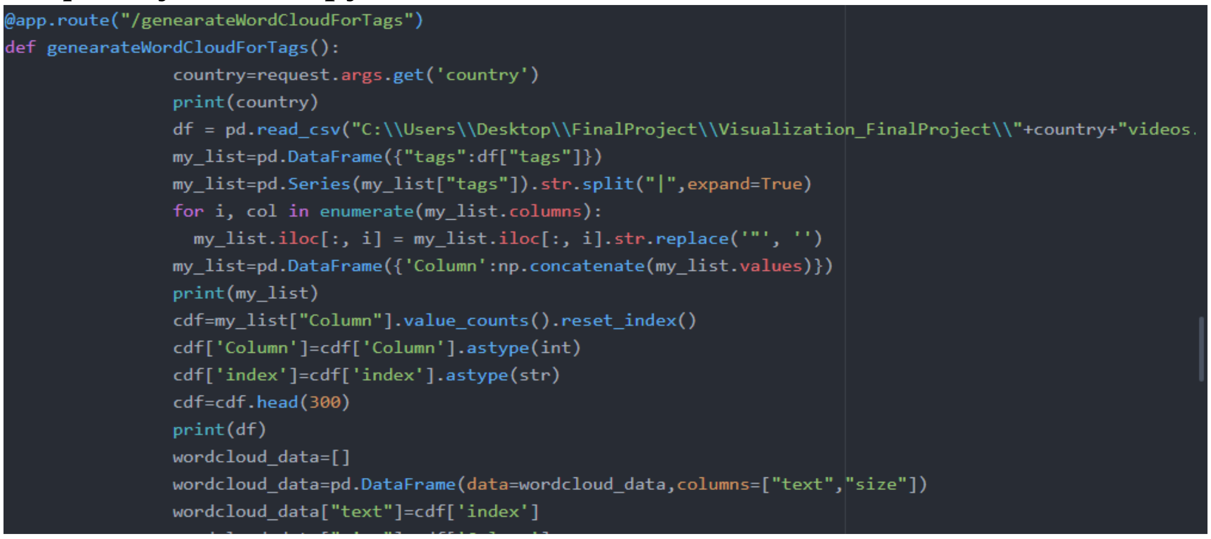


```
cell.filter(function(d) { return d.i === d.j; }).append("text")
    .attr("x", padding)
    .attr("y", padding)
    .attr("dy", ".71em")
    .text(function(d) { return d.x; });

cell.call(brush);

function plot(p) {
  var cell = d3v3.select(this);

  x.domain(domainByTrait[p.x]);
  y.domain(domainByTrait[p.y]);

  cell.append("rect")
      .attr("class", "frame")
      .attr("x", padding / 2)
      .attr("y", padding / 2)
      .attr("width", size - padding)
      .attr("height", size - padding);
```

Tags entered when a video is published is collected from the dataset and word cloud visualization is used to get insight on what are the words that are used most frequently when a video publishes is in trending list and the visualization for the same is as below:

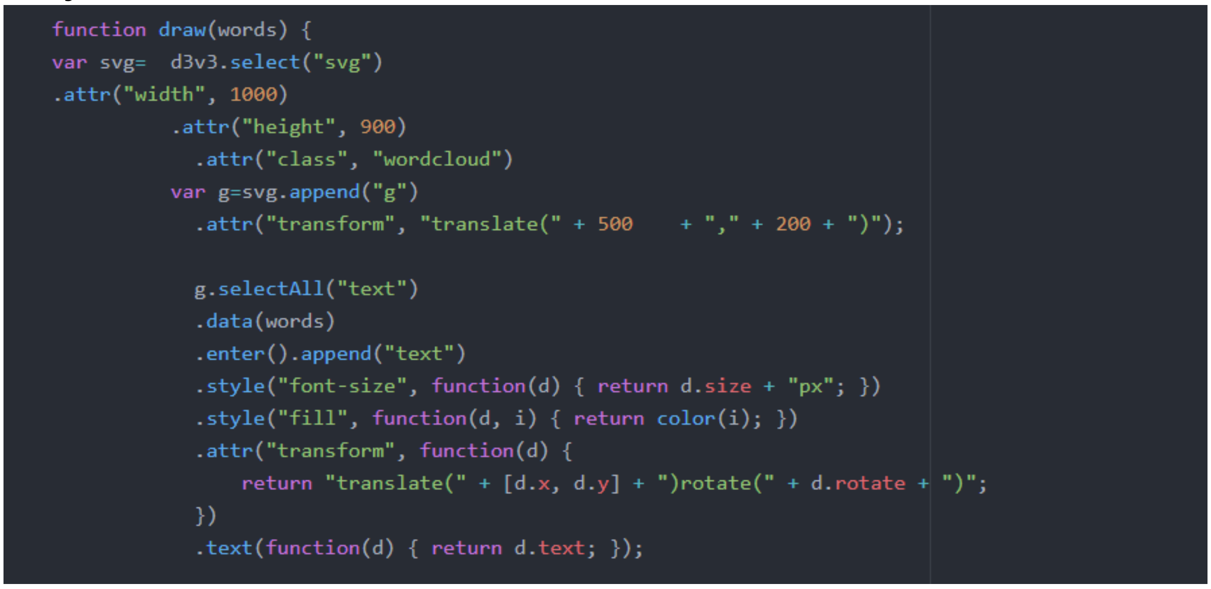


From above visualization we can say that when USA dataset is used tags like ‘**how to**’, ‘**Pop**’, ’**human**’ etc., are used most frequently and the python code for the same is as below:



```
@app.route("/genearateWordCloudForTags")
def genearateWordCloudForTags():
                country=request.args.get('country')
                print(country)
                df = pd.read_csv("C:\\Users\\Desktop\\FinalProject\\Visualization_FinalProject\\"+country+"videos.
                my_list=pd.DataFrame({"tags":df["tags"]})
                my_list=pd.Series(my_list["tags"]).str.split("|",expand=True)
                for i, col in enumerate(my_list.columns):
                  my_list.iloc[:, i] = my_list.iloc[:, i].str.replace('"', '')
                my_list=pd.DataFrame({'Column':np.concatenate(my_list.values)})
                print(my_list)
                cdf=my_list["Column"].value_counts().reset_index()
                cdf['Column']=cdf['Column'].astype(int)
                cdf['index']=cdf['index'].astype(str)
                cdf=cdf.head(300)
                print(df)
                wordcloud_data=[]
                wordcloud_data=pd.DataFrame(data=wordcloud_data,columns=["text","size"])
                wordcloud_data["text"]=cdf['index']
```

D3.js code for the above visualization is as below:



```
    function draw(words) {
    var svg=  d3v3.select("svg")
    .attr("width", 1000)
              .attr("height", 900)
                .attr("class", "wordcloud")
              var g=svg.append("g")
                .attr("transform", "translate(" + 500    + "," + 200 + ")");

                g.selectAll("text")
                .data(words)
                .enter().append("text")
                .style("font-size", function(d) { return d.size + "px"; })
                .style("fill", function(d, i) { return color(i); })
                .attr("transform", function(d) {
                    return "translate(" + [d.x, d.y] + ")rotate(" + d.rotate + ")";
                })
                .text(function(d) { return d.text; });
```

All the above visualizations used are obtained USA dataset and these visualizations varied for each country and in the video recording same is captured for different countries.

5.Conclusion

- We analysed the dataset of different countries and audience interests do vary from one country to another country like in USA Entertainment and Music stand in Top 2 categories, where as in India it is Entertainment and News & People which stand out as Top 2. Similarly each countries top 2 categories varied.
- For all the countries dataset the correlation between likes and views is always positive.
- We also made some predictions which include when is the correct time to publish a video and what's the time it takes for a video to get into trending list.
- From bubble chart we could see that all channel which come under entertainment and music category has more videos in the trending list, also here these insights vary with each country.

6.References

1 Dataset:https://github.com/DataSnaek/Trending-YouTube-Scraper
2 https://www.kaggle.com/datasnaek/youtube-new.
3 https://bl.ocks.org/mbostock/4063663
4 https://bl.ocks.org/alokkshukla/3d6be4be0ef9f6977ec6718b2916d168
5 http://bl.ocks.org/d3noob/8952219
6 https://blog.risingstack.com/d3-js-tutorial-bar-charts-with-javascript/
7 http://bl.ocks.org/ericcoopey/6382449
8 https://github.com/jasondavies/d3-cloud