# Ad Effectiveness: Analysis of Ad Interactivity and Location

Github link: https://github.com/hjeffreywang/Ad-Effectiveness-Report

---

=================================================================================================

## Work below

### Scenario

We are given data from ad sites such as google and facebook, how can these metrics be used to create actionable insights for the rest of the teams? Done in presentation form for ease of process.

### Goal

ABCJewelry wishes to increase sales while also reducing ad costs. Our job is to find actionable insights to help with those decisions.

### Key Tasks

1. Data Acquisition and Exploration
2. Feature Processing
3. Data Analysis
4. Data Engineering
5. Visualization Development

### Deliverables

1. How effective are our ads? (appearances, clicks, and Clicks/Appearance ratio)
2. Which ads are best? (Campaign, Site, Platform)
3. How long before ads become ineffective (# of appearances vs length of time)
4. Where are our audience? (Most campaigns, most appearances, most clicks)

---

## Feature definitions

- **campaign_item_id** : unique id of each adevertising campaign
- **no_of_days** : number of days campaign has been running
- **time** : timestamp on which the data was captured
- **ext_service_id** : id of each advertising platforms used
- **ext_service_name** : name of each advertising platforms used
- **creative_id** : id of the creative images used for ads
- **creative_height** : height of the creative image for the ad in pixels
- **creative_width** : width of the creative image for the ad in pixels
- **search_tags** : search tags used for displaying ads
- **template_id** : template used in the creative image
- **landing_page** : landing page url on which users clicked or browsed through
- **advertiser_id** : id of the advertiser
- **advertiser_name** : name of the place of the advertiser ( city , country , state )
- **network_id** : id of the each agency
- **advertiser_currency** : currency of the country in which the advertiser operates in
- **channel_id** : id of each channel used for placed ads
- **channel_name** : name of the channel ( display , search , social , mobile video )
- **max_bid_cpm** : maximum value of bid for optimizing cpm
- **campaign_budget_usd** : overall budget of the campaign or the amount of money that the campaign can spend
- **impressions** : the number of times an advertisement is displayed on a website or social media platform.
- **clicks** : the number of times an advertisement is clicked on by a user, leading them to the advertiser's website or landing page.
- **currency_code** : the currency code of the advertiser
- **exchange_rate** : a relative price of one currency expressed in terms of another currency.
- **media_cost_usd** : the amount of money that the campaign has spent on that particuar day
- **position_in_content** : position where the ad was placed on the website page
- **unique_reach** : the number of unique users who see your post or page.
- **total_reach** : the number of people who saw any content from your page or about your page.
- **search_tags** : a word or set of words a person enters when searching on Google or one of our Search Network sites.
- **cmi_currency_code** : campaign currency code
- **time_zone** : timezone in which the campaign is running
- **weekday_cat** : weekday / weekend catgeory
- **keywords** : a word or set of words that Google Ads advertisers can add to a given ad group so that your ads are targeting the right audience.

## Import Required Libraries

In [1]:
```python
# Import
import numpy as np
import pandas as pd
import altair as alt
import pandas_profiling as pp


import matplotlib.pyplot as plt
%matplotlib inline

import seaborn as sns

import warnings # to avoid warnings
warnings.filterwarnings('ignore')
```

```
/home/jeffwa/anaconda3/envs/DL_new/lib/python3.10/site-packages/tqdm/auto.py:22: TqdmWarning: IProgress not found. Please updat
e jupyter and ipywidgets. See https://ipywidgets.readthedocs.io/en/stable/user_install.html
  from .autonotebook import tqdm as notebook_tqdm
/tmp/ipykernel_22247/160099804.py:5: DeprecationWarning: `import pandas_profiling` is going to be deprecated by April 1st. Plea
se use `import ydata_profiling` instead.
  import pandas_profiling as pp
```

In [2]:
```python
import plotly.graph_objs as go
import plotly.offline as pyo
import plotly.express as px
```

In [3]:
```python
# display all columns of the dataframe
pd.options.display.max_columns = None

# use below code to convert the 'exponential' values to float
np.set_printoptions(suppress=True)
```

In [4]:
```python
# set the plot size using 'rcParams'
# once the plot size is set using 'rcParams', it sets the size of all the forthcoming plots in the file
# pass width and height in inches to 'figure.figsize'
plt.rcParams['figure.figsize'] = [15,8]
```

## Load and Exploring the dataset

In [5]:
```python
# load
df=pd.read_csv("dataset.csv",low_memory=False)

# preview 5 first 5 rows
df.head(5)
```

Out[5]:

| | campaign_item_id | no_of_days | time | ext_service_id | ext_service_name | creative_id | creative_width | creative_height | search_tags | template_id | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 2733 | 7 | 2022-05-01 | 128 | Facebook Ads | 1000 | 300.0 | 250.0 | #The Power of X | 90.0 | https://www.a /collec |
| **1** | 2733 | 8 | 2022-05-02 | 16 | DV360 | 1000 | 300.0 | 250.0 | #Be Bold. Be X | 90.0 | https://www.a /women/co |
| **2** | 2733 | 9 | 2022-05-03 | 128 | Facebook Ads | 1000 | 300.0 | 250.0 | #Embrace Your Individuality with X | 90.0 | https://www.a /collec |
| **3** | 2733 | 10 | 2022-05-04 | 128 | Facebook Ads | 1000 | 300.0 | 250.0 | #Be Bold. Be X | 90.0 | https://www.a /colle |
| **4** | 2733 | 11 | 2022-05-05 | 4 | Google Ads | 1000 | 300.0 | 250.0 | #Be Bold. Be X | 90.0 | https://www.a /collec |

In [6]:
```python
# see total number of rows
df.shape
```

Out[6]:
```
(72612, 35)
```

In [7]:
```python
df.columns
```

Out[7]:
```
Index(['campaign_item_id', 'no_of_days', 'time', 'ext_service_id',
       'ext_service_name', 'creative_id', 'creative_width', 'creative_height',
       'search_tags', 'template_id', 'landing_page', 'advertiser_id',
       'advertiser_name', 'network_id', 'approved_budget',
       'advertiser_currency', 'channel_id', 'channel_name', 'max_bid_cpm',
       'network_margin', 'campaign_budget_usd', 'impressions', 'clicks',
       'stats_currency', 'currency_code', 'exchange_rate', 'media_cost_usd',
       'position_in_content', 'unique_reach', 'total_reach', 'search_tag_cat',
       'cmi_currency_code', 'timezone', 'weekday_cat', 'keywords'],
      dtype='object')
```

In [8]:
```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 72612 entries, 0 to 72611
Data columns (total 35 columns):
 #   Column               Non-Null Count  Dtype  
---  ------               --------------  -----  
 0   campaign_item_id     72612 non-null  int64  
 1   no_of_days           72612 non-null  int64  
 2   time                 72612 non-null  object 
 3   ext_service_id       72612 non-null  int64  
 4   ext_service_name     72612 non-null  object 
 5   creative_id          72612 non-null  int64  
 6   creative_width       69200 non-null  float64
 7   creative_height      69200 non-null  float64
 8   search_tags          72612 non-null  object 
 9   template_id          69200 non-null  float64
 10  landing_page         72612 non-null  object 
 11  advertiser_id        72612 non-null  int64  
 12  advertiser_name      72612 non-null  object 
 13  network_id           72612 non-null  int64  
 14  approved_budget      72206 non-null  float64
 15  advertiser_currency  72612 non-null  object 
 16  channel_id           72612 non-null  int64  
 17  channel_name         72612 non-null  object 
 18  max_bid_cpm          7406 non-null   float64
 19  network_margin       72612 non-null  float64
 20  campaign_budget_usd  72612 non-null  float64
 21  impressions          72612 non-null  int64  
 22  clicks               72612 non-null  int64  
 23  stats_currency       72612 non-null  object 
 24  currency_code        72612 non-null  object 
 25  exchange_rate        72612 non-null  int64  
 26  media_cost_usd       72612 non-null  float64
 27  position_in_content  0 non-null      float64
 28  unique_reach         0 non-null      float64
 29  total_reach          0 non-null      float64
 30  search_tag_cat       72612 non-null  object 
 31  cmi_currency_code    72612 non-null  object 
 32  timezone             72612 non-null  object 
 33  weekday_cat          72612 non-null  object 
 34  keywords             72612 non-null  object 
dtypes: float64(11), int64(10), object(14)
memory usage: 19.4+ MB
```

## Cleaning Null Values

In [9]:
```
# sort the variables on the basis of total null values in the variable
Total = df.isnull().sum().sort_values(ascending = False)

#calculate nulls
Percent = (df.isnull().sum()*100/df.isnull().count()).sort_values(ascending = False)
missing_data = pd.concat([Total, Percent], axis = 1, keys = ['Total', 'Percentage of Missing Values'])

# add the column containing data type of each variable
missing_data['Type'] = df[missing_data.index].dtypes
missing_data
```

Out[9]:

| | Total | Percentage of Missing Values | Type |
|---|---|---|---|
| **unique_reach** | 72612 | 100.000000 | float64 |
| **total_reach** | 72612 | 100.000000 | float64 |
| **position_in_content** | 72612 | 100.000000 | float64 |
| **max_bid_cpm** | 65206 | 89.800584 | float64 |
| **creative_width** | 3412 | 4.698948 | float64 |
| **creative_height** | 3412 | 4.698948 | float64 |
| **template_id** | 3412 | 4.698948 | float64 |
| **approved_budget** | 406 | 0.559136 | float64 |
| **exchange_rate** | 0 | 0.000000 | int64 |
| **clicks** | 0 | 0.000000 | int64 |
| **stats_currency** | 0 | 0.000000 | object |
| **currency_code** | 0 | 0.000000 | object |
| **campaign_item_id** | 0 | 0.000000 | int64 |
| **media_cost_usd** | 0 | 0.000000 | float64 |
| **campaign_budget_usd** | 0 | 0.000000 | float64 |
| **search_tag_cat** | 0 | 0.000000 | object |
| **cmi_currency_code** | 0 | 0.000000 | object |
| **timezone** | 0 | 0.000000 | object |
| **weekday_cat** | 0 | 0.000000 | object |
| **impressions** | 0 | 0.000000 | int64 |
| **channel_name** | 0 | 0.000000 | object |
| **network_margin** | 0 | 0.000000 | float64 |
| **no_of_days** | 0 | 0.000000 | int64 |
| **channel_id** | 0 | 0.000000 | int64 |
| **advertiser_currency** | 0 | 0.000000 | object |
| **network_id** | 0 | 0.000000 | int64 |
| **advertiser_name** | 0 | 0.000000 | object |
| **advertiser_id** | 0 | 0.000000 | int64 |
| **landing_page** | 0 | 0.000000 | object |
| **search_tags** | 0 | 0.000000 | object |
| **creative_id** | 0 | 0.000000 | int64 |
| **ext_service_name** | 0 | 0.000000 | object |
| **ext_service_id** | 0 | 0.000000 | int64 |
| **time** | 0 | 0.000000 | object |
| **keywords** | 0 | 0.000000 | object |

In [10]:
```
# creative width
df['creative_width'] = df['creative_width'].fillna(0)

# creative height
df['creative_height'] = df['creative_height'].fillna(0)

# template id
df['template_id'] = df['template_id'].fillna(-1)

# approved_budget
df['approved_budget'] = df['approved_budget'].fillna(0)
```

## Drop unnecessary columns

- Prune features that are entirely made up of null or actively harmful to analysis.

In [11]:
```
df.drop(columns=['position_in_content','unique_reach','total_reach','max_bid_cpm'],inplace=True)
```

- **no_of_days** : campaigns run for atleast a month , so when no_of_days == 0 means one day only.

```
In [12]: df.describe()
```

Out[12]:

| | campaign_item_id | no_of_days | ext_service_id | creative_id | creative_width | creative_height | template_id | advertiser_id | network_id | approved_budget |
|---|---|---|---|---|---|---|---|---|---|---|
| **count** | 72612.000000 | 72612.000000 | 72612.000000 | 72612.000000 | 72612.000000 | 72612.000000 | 72612.000000 | 72612.000000 | 72612.000000 | 7.261200e+04 |
| **mean** | 3130.143282 | 27.036344 | 49.449127 | 7450.124842 | 255.226409 | 212.688674 | 79.131659 | 6195.862213 | 345.272861 | 1.251030e+05 |
| **std** | 142.154918 | 25.479175 | 55.881324 | 4062.384982 | 106.899767 | 89.083139 | 27.033401 | 387.864576 | 23.707191 | 5.611237e+05 |
| **min** | 2733.000000 | 0.000000 | 4.000000 | 1000.000000 | 0.000000 | 0.000000 | -1.000000 | 4756.000000 | 188.000000 | 0.000000e+00 |
| **25%** | 3148.000000 | 9.000000 | 4.000000 | 3725.000000 | 300.000000 | 250.000000 | 90.000000 | 6319.000000 | 353.000000 | 6.000000e+03 |
| **50%** | 3173.000000 | 19.000000 | 16.000000 | 7855.000000 | 300.000000 | 250.000000 | 90.000000 | 6385.000000 | 353.000000 | 1.000000e+04 |
| **75%** | 3202.000000 | 37.000000 | 128.000000 | 10995.000000 | 300.000000 | 250.000000 | 90.000000 | 6394.000000 | 353.000000 | 1.500000e+04 |
| **max** | 3960.000000 | 118.000000 | 128.000000 | 15605.000000 | 300.000000 | 250.000000 | 93.000000 | 6490.000000 | 353.000000 | 6.000000e+06 |

- **ext_service_name** : most ads were Facebook Ads since it is the most populated social channel for target audience.
- **landing_page** : boho jewelry page has the most clicked ads.

```
In [13]: # summary of categorical variables
         df.describe(include=object)

         # Note: If we pass 'include=object' to the .describe(), it will return descriptive statistics for categorical variables only
```

Out[13]:

| | time | ext_service_name | search_tags | landing_page | advertiser_name | advertiser_currency | channel_name | stats_currency | currency_code |
|---|---|---|---|---|---|---|---|---|---|
| **count** | 72612 | 72612 | 72612 | 72612 | 72612 | 72612 | 72612 | 72612 | 72612 |
| **unique** | 224 | 3 | 6 | 45 | 44 | 5 | 5 | 5 | 5 |
| **top** | 2022-10-22 | Facebook Ads | #The Ultimate Fashion Statement with X | https://www.abcjewelry.com /collections/boho-je... | Oman | AED | Mobile | AED | AED |
| **freq** | 955 | 24275 | 12293 | 1684 | 8641 | 53661 | 14625 | 53661 | 53661 |

### Creating a metric to measure Clicks per appearance

```
In [14]: df['ctr']=(df['clicks']/df['impressions'])*100
```

```
In [15]: df['ext_service_name'].value_counts()
```

```
Out[15]: Facebook Ads    24275
         DV360           24171
         Google Ads      24166
         Name: ext_service_name, dtype: int64
```

# Visualization Implementations

**Reasonings are conveyed at the chart descriptions**

## Histogram chart

A histogram is used to illustrate the distribution of a dataset and displays which values are most frequent.

**Reasons**

1. To calculate the probability of representation of any value of a continuous variable
2. Helps to visualize whether the distribution is symmetric or skewed left or right.
3. It can also show any outliers or gaps in the data.

---

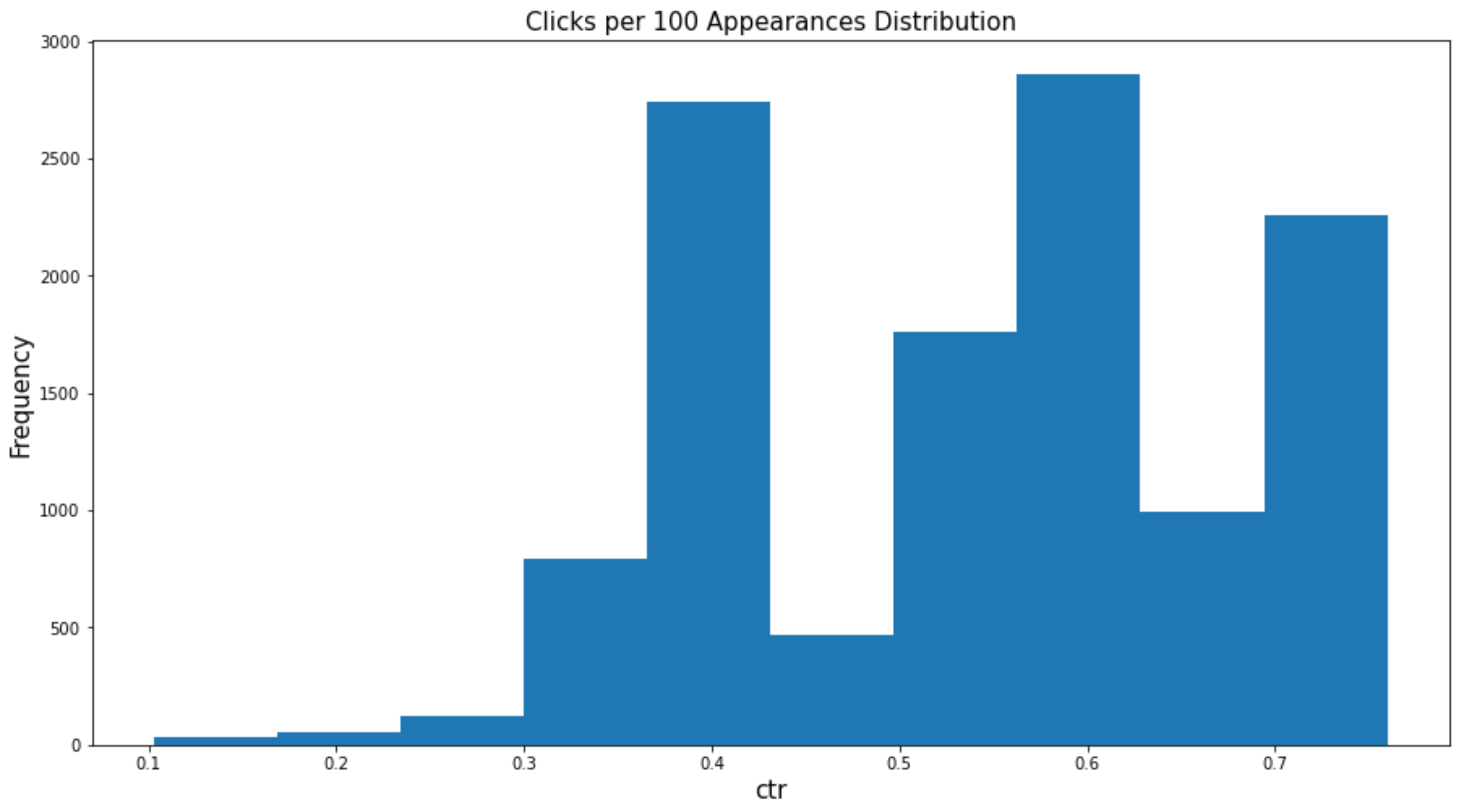
The benchmarks for CTR ( Click through rate ) is 0.76% for Style & Fashion tags (Google) and 2.71% to be in Top 10% competition. Our CTR distribution lies between 0.76-2.71 for ABC company.

```
In [16]: # Clicks per 100 Appearances Frequency Distribution
         # set the xlabel and the fontsize
         plt.xlabel("ctr", fontsize=15)

         # set the ylabel and the fontsize
         plt.ylabel("Frequency", fontsize=15)

         # set the title of the plot
         plt.title("Clicks per 100 Appearances Distribution", fontsize=15)

         # plot the histogram for the target variable
         plt.hist(df.loc[(df["ctr"]>=0.1) & (df["ctr"]<=0.76)]['ctr'])
         plt.show()
```

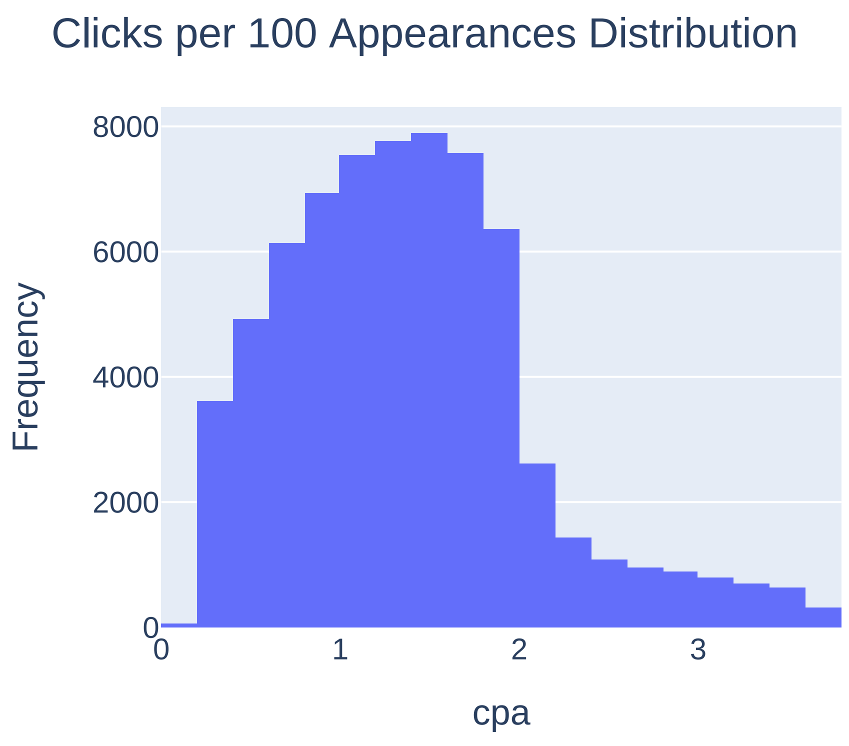


```
In [17]: # Filter the data and create a histogram
hist_data = df.loc[(df["ctr"]>0.1) & (df["ctr"]<=3.71)]['ctr']
fig = px.histogram(hist_data, nbins=20)

# Set the layout properties
fig.update_layout(
    title="Clicks per 100 Appearances Distribution",
    xaxis_title="cpa",
    yaxis_title="Frequency",
    font=dict(size=15),
    showlegend=False,    width=500, # set width to 500 pixels
    height=400, # set height to 500 pixels

)

fig.show()
```



## SCATTER PLOT

Purpose:

1. Identify easily visible patterns and relationships

2. Detecting outliers

3. Visualize trends over time

---

## Conclusions

1. As campaign length increases, impressions and clicks decrease

2. Campaigns of longer duration have constant & low impressions and clicks

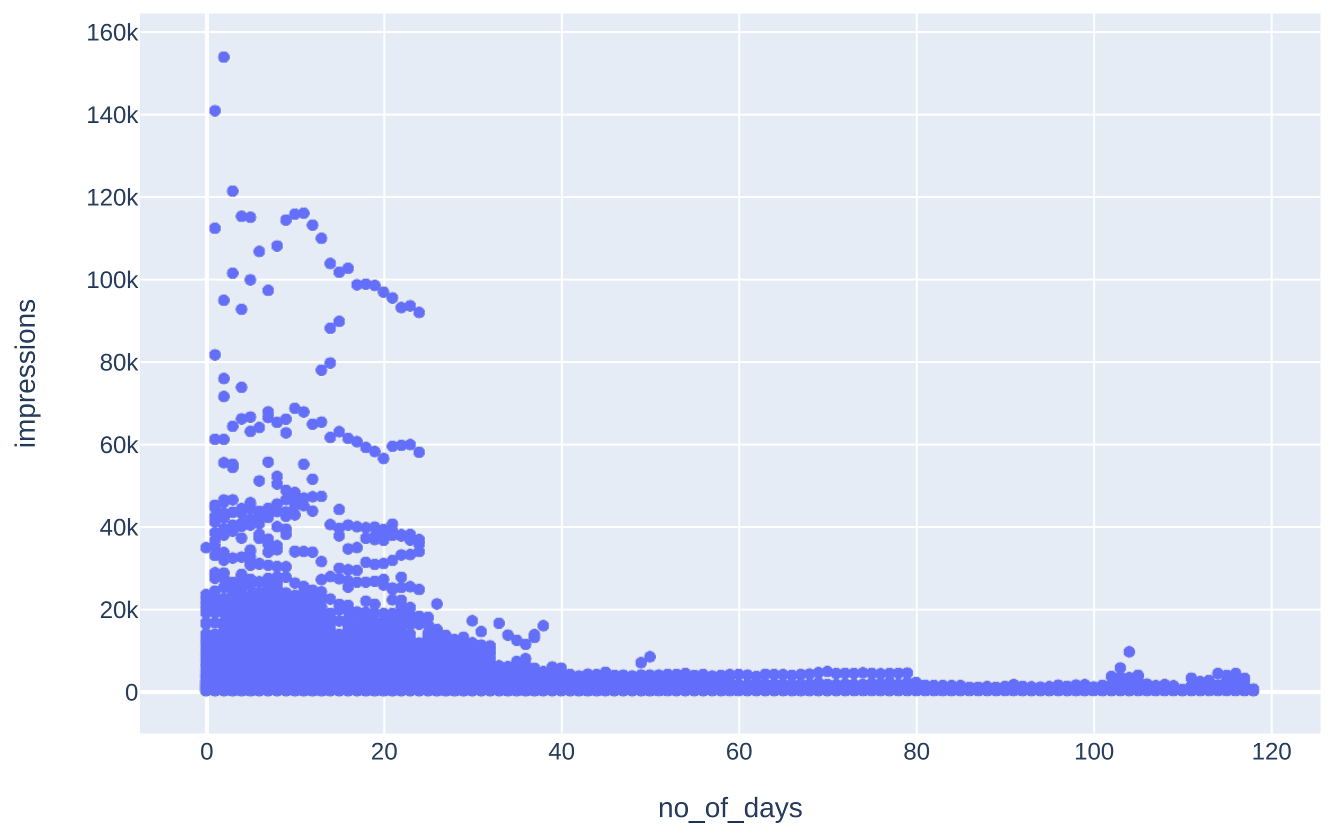
3. The graphs below can show that most campaigns with less duration were newly created or paused due to poor performance.

4. We can spot outliers in both the graphs which indicates sudden spikes in the impressions and clicks which maybe due to certain events such as festivals , social media popularity , etc . We can further analyse at what time of the day , on which days , in which season , festivals , national or public holidays the performance usually goes up.

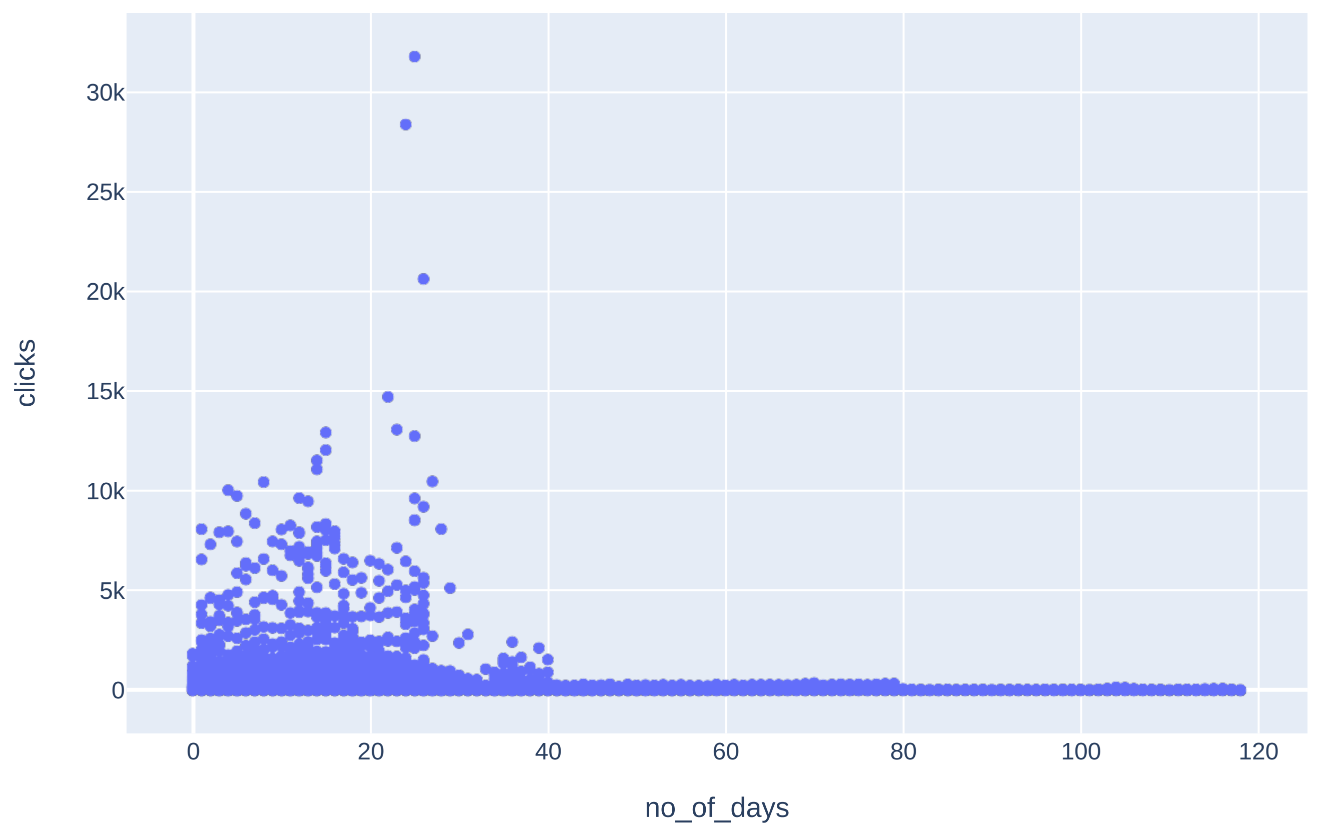
## Prettier Graph

In [18]:
```
import plotly.express as px

fig = px.scatter(df, x="no_of_days", y="impressions", trendline="ols",    width=750, # set width to 500 pixels
    height=500, # set height to 500 pixels
)
fig.show()
```



In [19]:
```
fig = px.scatter(df, x="no_of_days", y="clicks", trendline="ols",    width=750, # set width to 500 pixels
    height=500, # set height to 500 pixels
)
fig.show()
```

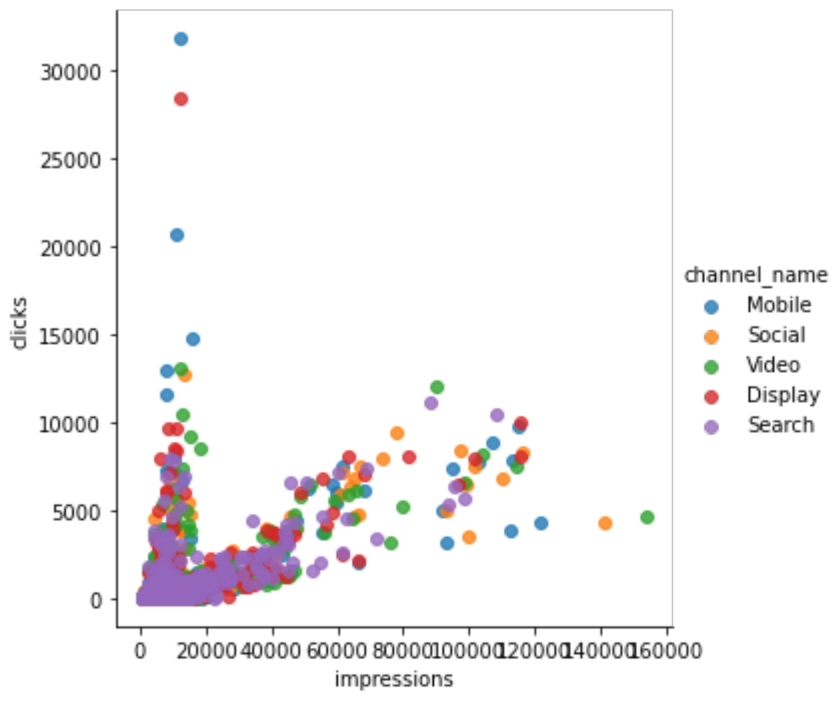


## Conclusions

1. Mobile campaigns are able to get higher clicks
2. The performance of search campaigns needs to be improved
3. Social campaigns are able to reach audiences more but unable to get conversions

```
In [20]: # scatter plot : impressions vs clicks ( hue : channel_name )
         sns.lmplot(x = "impressions", y = "clicks", data = df, fit_reg=False, hue='channel_name')
```

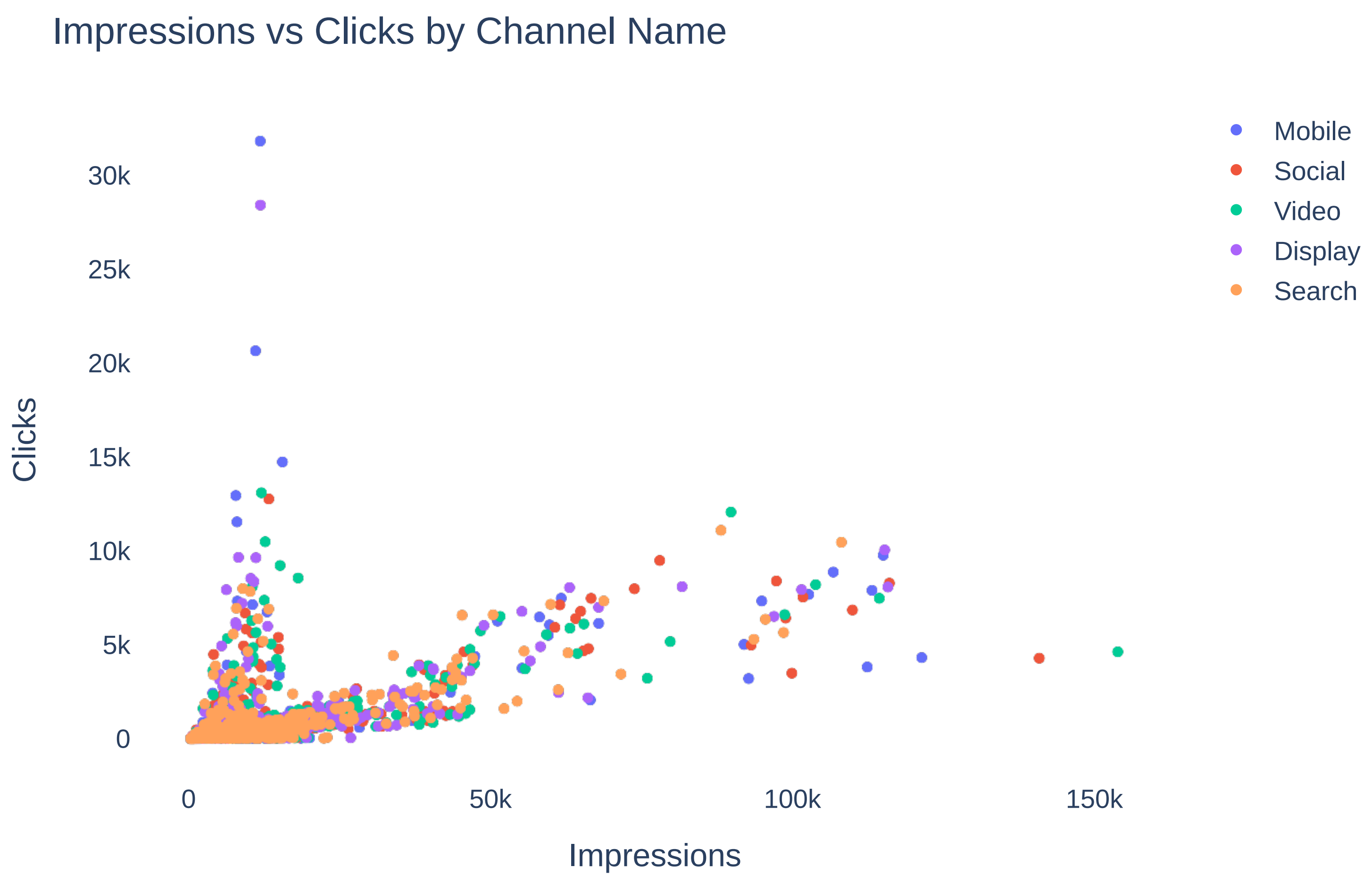
```
Out[20]: <seaborn.axisgrid.FacetGrid at 0x7fc95ea4c3d0>
```



```
In [21]: fig = px.scatter(df, x="impressions", y="clicks", color="channel_name",
                          hover_name="channel_name",
                          labels={"impressions": "Impressions", "clicks": "Clicks"},
                          title="Impressions vs Clicks by Channel Name")

         fig.update_layout(
             font=dict(size=14),
             legend=dict(title=None),
             plot_bgcolor="white",
             margin=dict(l=80, r=20, t=60, b=80),
             width=750, # set width to 500 pixels
             height=500, # set height to 500 pixels
         )


         fig.show()
```



## Conclusions

1. Top 3 countries where the campaigns are running are India, Oman , Qatar & UAE. This means the our company mainly operates in the Middle Eastern Asian region .
2. We can further find out performance metrics of each country vs budget they were alloted to gauge the relative performance and take more informed decision.

## Making a more Readable Visualization

We need to group Indian states as one country.

In [22]:
```
#  calculate data
labels=df['advertiser_name'].value_counts().index,
values=df['advertiser_name'].value_counts(),
```

In [23]:
```
df_abridged=df
```

In [24]:
```
df.head()
```

Out[24]:
```
   campaign_item_id  no_of_days        time  ext_service_id  ext_service_name  creative_id  creative_width  creative_height  search_tags  template_id
0              2733           7  2022-05-01             128      Facebook Ads         1000           300.0            250.0  #The Power          90.0  https://www.a
                                                                                                                                  of X                  /collec
1              2733           8  2022-05-02              16             DV360         1000           300.0            250.0  #Be Bold. Be        90.0  https://www.a
                                                                                                                                     X                  /women/co
                                                                                                                              #Embrace
2              2733           9  2022-05-03             128      Facebook Ads         1000           300.0            250.0     Your             90.0  https://www.a
                                                                                                                           Individuality                /collec
                                                                                                                                with X
3              2733          10  2022-05-04             128      Facebook Ads         1000           300.0            250.0  #Be Bold. Be        90.0  https://www.a
                                                                                                                                     X                  /collec
4              2733          11  2022-05-05               4        Google Ads         1000           300.0            250.0  #Be Bold. Be        90.0  https://www.a
                                                                                                                                     X                  /collec
```

In [25]:
```
India = ['Andhra Pradesh', 'Karnataka', 'Pan India','North', 'Gujarat', 'Orissa', 'Tamil Nadu', 'Maharashtra', 'West Bengal', '

Oman = ['Muscat','Sohar']

Qatar = ['Doha']

UAE = ['Dubai', 'Abu Dhabi']

Bahrain = ['Manama']

Kuwait = ['Kuwait City', 'Al Ahmadi']

KSA = ['Jeddah']

Malaysia = ['Kuala Lumpur']

Singapore = ['Singapore']

USA = ['New York']

Thailand = ['Bangkok']

Egypt = ['Cairo', 'Luxor',  'Almaza Bay']

Bangladesh = ['Chattogram', 'Chandpur']

Ethiopia = ['Addis Ababa']
```

In [26]:
```
cities_by_country = {'India': India,
'Oman': Oman,
'Qatar': Qatar,
'UAE': UAE,
'Bahrain': Bahrain,
'Kuwait': Kuwait,
'KSA': KSA,
'Malaysia': Malaysia,
'Singapore': Singapore,
'USA': USA,
'Thailand': Thailand,
'Egypt': Egypt,
'Bangladesh': Bangladesh,
'Ethiopia': Ethiopia}
```

In [27]:
```
for country,cities in cities_by_country.items():
    df.loc[df['advertiser_name'].isin(cities), "advertiser_name"] = country
```

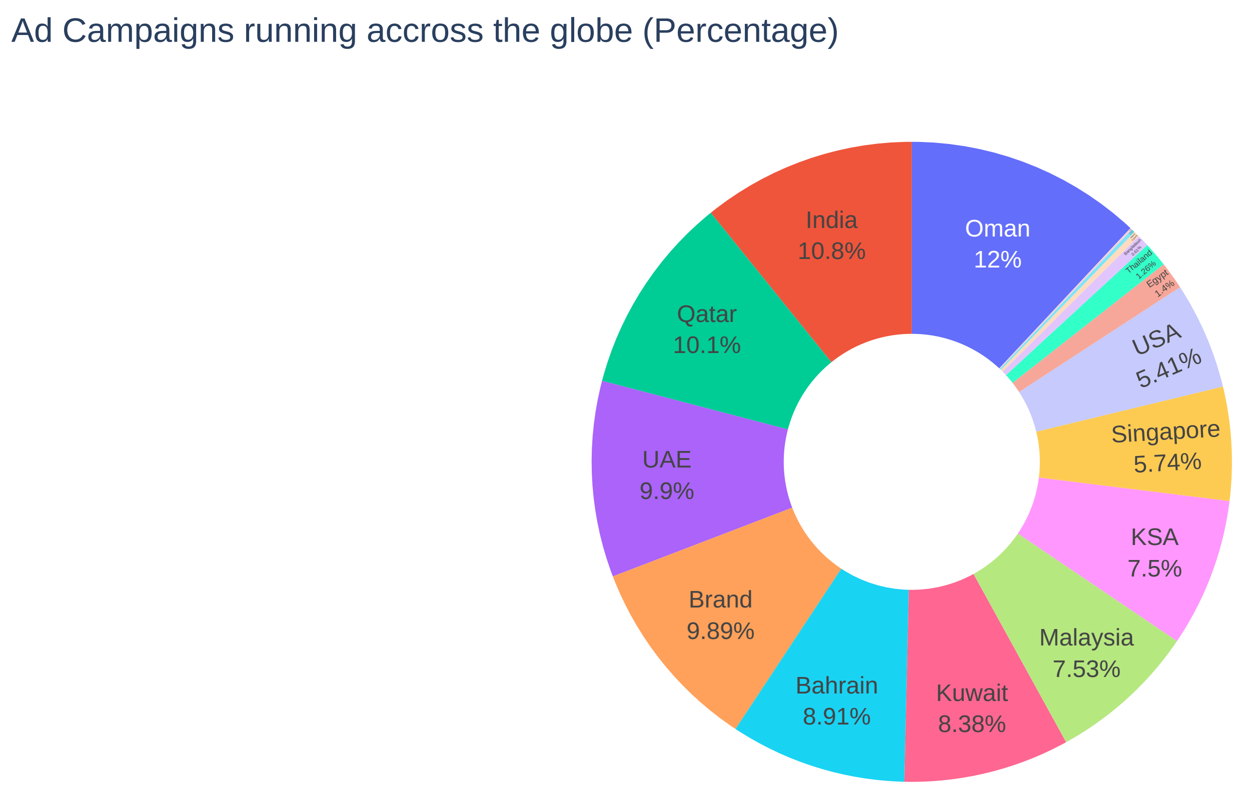
In [28]:
```
label = df['advertiser_name'].value_counts().index
name = df['advertiser_name'].value_counts()
```

In [29]:
```python
# Create data for the Pie Chart
data = [go.Pie(labels=name.index,
               values=name,
               hole=0.4,
               textposition='inside',
               textinfo='label+percent',
               hoverinfo='label+percent+value')]

# Set layout for the Pie Chart
layout = go.Layout(title='Ad Campaigns running accross the globe (Percentage)',
                   showlegend=False,
                   legend=dict(orientation="h"),
                   width=1000,
                   height=500,)

# Create figure object
fig = go.Figure(data=data, layout=layout)

# Display the figure
pyo.iplot(fig)
```



In [30]:
```python
data_2=df.groupby('advertiser_name')['clicks'].sum().sort_values()
```

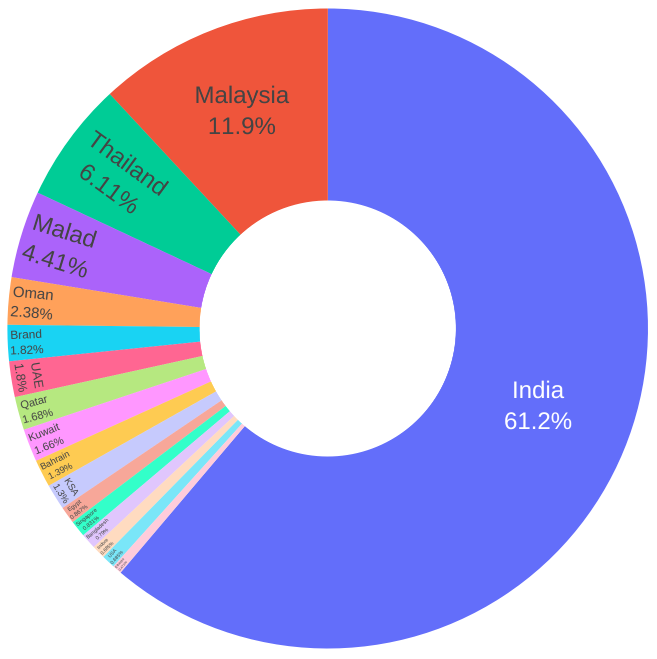
In [31]:
```python
# Create data for the Pie Chart
data = [go.Pie(labels=data_2.index,
               values=data_2,
               hole=0.4,
               textposition='inside',
               textinfo='label+percent',
               hoverinfo='label+percent+value')]

# Set layout for the Pie Chart
layout = go.Layout(title='Clicks across the Globe (%)',
                   showlegend=False,
                    width=1000,
                   height=500,)

# Create figure object
fig = go.Figure(data=data, layout=layout)

# Display the figure
pyo.iplot(fig)
```

Clicks across the Globe (%)



```
In [32]: data_3=df.groupby('advertiser_name')['impressions'].sum().sort_values()
```

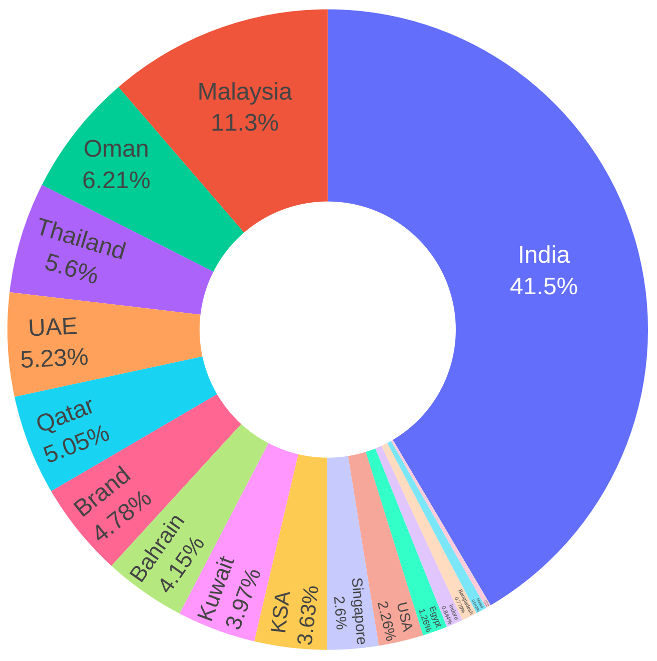
```
In [33]: # Create data for the Pie Chart
         data = [go.Pie(labels=data_3.index,
                        values=data_3,
                        hole=0.4,
                        textposition='inside',
                        textinfo='label+percent',
                        hoverinfo='label+percent+value')]

         # Set layout for the Pie Chart
         layout = go.Layout(title='Ad Appearances across the Globe (%)',
                            showlegend=False,
                             width=1000,
                            height=500,)

         # Create figure object
         fig = go.Figure(data=data, layout=layout)

         # Display the figure
         pyo.iplot(fig)
```

Ad Appearances across the Globe (%)



```
In [34]: data_4=df.groupby('advertiser_name')['ctr'].mean().sort_values()
```