Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

Department of Computer Engineering

Name of the Student: Shashwat Shah **Roll Number: O201**

SAP ID: 60004220126 **Class: C2** **Division:C2** **Batch:C22**

Subject: WEB INTELLIGENCE

DATE OF PERFORMANCE: 03/04/2025 **DATE OF SUBMISION: 05/05/2025**

EXPERIMENT NO: 07

AIM:
Using Google Analytics, perform Audience Analysis, Acquisition Analysis, Behavior Analysis, Conversion Analysis.

THEORY:

Audience Analysis

Audience analysis is the method of obtaining information about the people in one's audience to better understand their wants, needs, values, and attitudes. It is the best if one first defines one's target audience by determining their demographics, such as their gender, age, and where they live. After the potential audience is defined, one will be able to narrow the list down to a single target group for analysis. Companies now have easy access to a wide range of audience information via social media channels. If one can turn all that data into clear insights about one's audience and what they care about, one can design a better business strategy.

Uses of audience analysis in business

1. Customer behaviour
2. Taking risks

Behavioural Analysis

Behaviour analysis is based on the foundations and principles of behaviourism. Behaviourism is a branch of psychology rooted in the idea that all behaviours result from conditioning processes. This branch of psychology focuses on understanding how associations, reinforcement, and punishment can be used to shape human behaviour. Behaviour is the result of circumstances. Behaviour analysis seeks to understand the impact of the events that come immediately after a behaviour. This understanding can be a useful tool for modifying problematic behaviours and teaching more adaptive responses. These strategies can be used in a wide variety of situations to help children and adults make positive changes in their lives.

Behaviour analysis can be a particularly effective learning tool for helping children with autism or developmental delays acquire and maintain new skills. Applied behavioural analysis therapy is a specific approach frequently used to treat autism and other conditions.

Conversion analysis

Conversion analysis is a method of tracking your website visitors' actions to see whether they do what you hope they will convertor drop off. A conversion analysis can help you identify which kinds of customers and which types of actions correspond to conversion, as well as when and where users fail to convert. It measures the impact of your experiments and tells you what's working to increase conversions, and what isn't.

To conduct a conversion analysis, you must understand your customers' journey. The best way to do this is to map out each step a customer takes in a conversion funnel so that you have an overview of the ideal steps needed to convert your website visitors into customers.

You can conduct a conversion analysis on any part of your product or website where you expect users to convert. This allows you to:

1. Understand a companies' conversion efficiency and scale over time.
2. Benchmark your conversion data against the competitors and the industry average.
3. Identify efficient marketing channels to uncover companies' go-to-market strategy and understand the ROI of their marketing spend.
4. Reveal the category performance of top retailers like Amazon, Walmart, and Target.
5. Gain visibility into what consumers are searching for within websites and how well different keywords convert.
6. Drill into consumer journey behaviors with on-site search and marketing channel conversion data to improve your customer acquisition and digital marketing strategy.

Google Analytics Dataset

The Google Merchandise Store sells Google branded merchandise. The data is typical of what you would see for an ecommerce website.

The sample dataset contains Google Analytics 360 data from the Google Merchandise Store, a real ecommerce store. The Google Merchandise Store sells Google branded merchandise. The data is typical of what you would see for an ecommerce website. It includes the following kinds of information:

Traffic source data: information about where website visitors originate. This includes data about organic traffic, paid search traffic, display traffic, etc.

Content data: information about the behavior of users on the site. This includes the URLs of pages that visitors look at, how they interact with content, etc.

Experiment 7

April 18, 2023

[ ]:
```
import numpy as np
import pandas as pd

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

[ ]:
```
import bq_helper
import matplotlib
import matplotlib.pyplot as plt
import seaborn as sns
from IPython.display import set_matplotlib_formats
from plotly.offline import init_notebook_mode, iplot
init_notebook_mode(connected=True)
import plotly.graph_objs as go
import plotly.express as px

set_matplotlib_formats('retina')
%matplotlib inline

from google.cloud import bigquery
client = bigquery.Client()
```

```
Using Kaggle's public dataset BigQuery integration.
```

1 Exploratory Data Analysis and Inference

[ ]:
```
query = '''
    SELECT
        FORMAT("%'d",COUNT(DISTINCT fullVisitorId)) AS users,
        FORMAT("%'d",SUM(totals.visits)) AS visits,
        FORMAT("%'d",SUM(totals.pageviews)) AS pageviews,
        FORMAT("%'d", SUM(totals.transactions)) AS transactions,
        SUM(totals.transactionRevenue)/1000000 AS revenue

    FROM
```

```
        `bigquery-public-data.google_analytics_sample.ga_sessions_*`
    WHERE
        _TABLE_SUFFIX BETWEEN '20161001' AND '20161230'
        AND totals.totalTransactionRevenue IS NOT NULL
'''

safe_query_job = client.query(query)
high_level_aug = safe_query_job.to_dataframe()

high_level_aug.head()
```

[ ]:

	users	visits	pageviews	transactions	revenue
0	2,868	3,176	92,822	3,314	386199.7

Inference : During 2016Q4, the Google merchendise store had over 3000 visits, almost 93,000 pageviews and earned $ 386,200 in revenue.

[ ]:
```
query = """
    SELECT
        DATE_TRUNC(PARSE_DATE('%Y%m%d',date), MONTH) AS month,
        SUM(totals.visits) AS visits,
        SUM(totals.transactionRevenue)/1000000 AS revenue

    FROM
        `bigquery-public-data.google_analytics_sample.ga_sessions_*`
    WHERE
        _TABLE_SUFFIX BETWEEN '20161001' AND '20161230'
        AND totals.totalTransactionRevenue IS NOT NULL
    GROUP BY 1
    ORDER BY 1

"""
safe_query_job = client.query(query)
df1 = safe_query_job.to_dataframe()
df1.head()
```

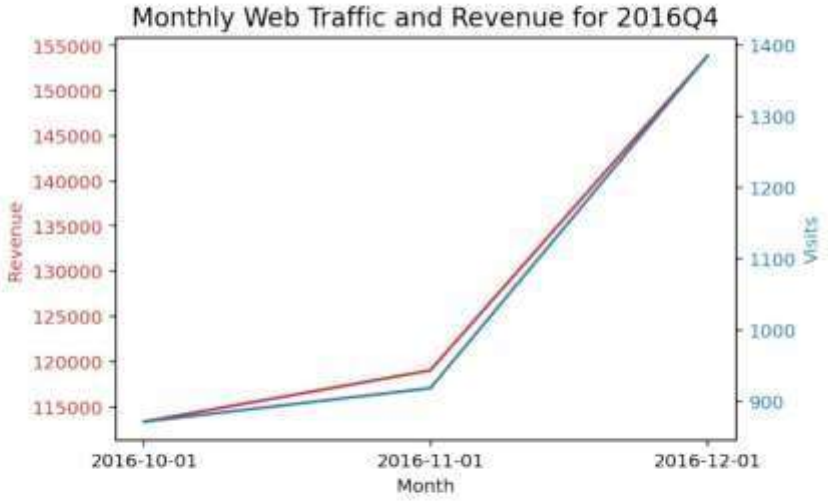
[ ]:

	month	visits	revenue
0	2016-10-01	872	113329.07
1	2016-11-01	919	119013.87
2	2016-12-01	1385	153856.76

1.0.1 Inference:

December generated over $ 153,000 in revenue and garned almost 1400 visits to the Google Merchandise store. The revenue generated in Dec accounted for 40% of Q4 sales, indicating a holiday surge that is typically seen in retail.

[ ]:
```
fig,ax1 = plt.subplots()
color = 'tab:red'
ax1.set_xlabel('Month')
ax1.set_ylabel('Revenue', color=color)
ax1.plot(df1['month'], df1['revenue'], color=color)
ax1.tick_params(axis='y', labelcolor=color)

ax2 = ax1.twinx()

color = 'tab:blue'
ax2.set_ylabel('Visits', color=color)
ax2.plot(df1['month'], df1['visits'], color=color)
ax2.tick_params(axis='y', labelcolor=color)

plt.title('Monthly Web Traffic and Revenue for 2016Q4', fontsize=14)
plt.xticks(df1['month'],rotation=45)
plt.show()
```



[ ]:
```
query = """
    SELECT
        geoNetwork.country AS country,
        SUM(totals.transactionRevenue)/1000000 AS revenue

    FROM
        `bigquery-public-data.google_analytics_sample.ga_sessions_*`
```

```
    WHERE
        _TABLE_SUFFIX BETWEEN '20161001' AND '20161230'
        AND totals.totalTransactionRevenue IS NOT NULL
    GROUP BY 1
    ORDER BY 2 desc

"""
safe_query_job = client.query(query)
df2 = safe_query_job.to_dataframe()
df2.head()
```

[ ]:

	country	revenue
0	United States	365518.96
1	Canada	9975.27
2	Kenya	3679.30
3	Puerto Rico	656.16
4	United Kingdom	527.10

[ ]:
```
fig = go.Figure(data=go.Choropleth(
    locations=df2['country'],
    z = df2['revenue'].astype(float),
    locationmode = 'country names',
    colorscale = 'Reds',
    colorbar_title = "revenue USD",
))

fig.update_layout(
    title_text = '2016Q4 Google Merchandise Store by Geo Location',)
fig.show()
```

1.1 Top channels sending traffic to GM

[ ]:
```
query = """
    SELECT
        channelGrouping as channel,
        SUM(totals.totalTransactionRevenue)/1000000 AS revenue,
        SUM(totals.transactions) AS transactions,
        COUNT(DISTINCT fullVisitorId) AS users,
        SUM(totals.visits) AS sessions,
        SUM(totals.pageviews) AS pageviews
    FROM
        `bigquery-public-data.google_analytics_sample.ga_sessions_*`
    WHERE
        _TABLE_SUFFIX BETWEEN '20161001' AND '20161230'
        AND totals.totalTransactionRevenue IS NOT NULL
    GROUP BY
        1
    ORDER BY
        2 DESC
"""
safe_query_job = client.query(query)
df3 = safe_query_job.to_dataframe()
df3.head()
```

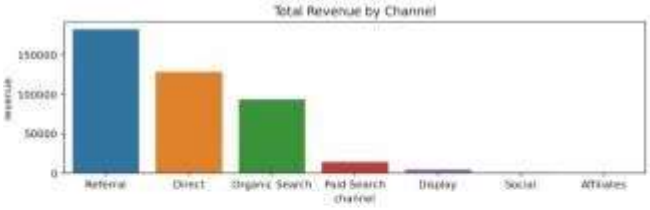
[ ]:

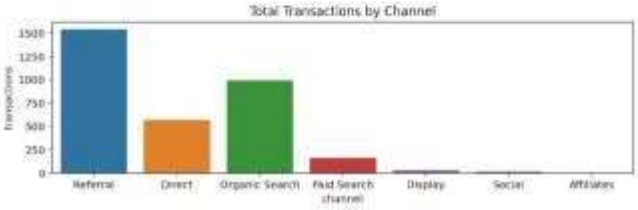
	channel	revenue	transactions	users	sessions	pageviews
0	Referral	182578.60	1540	1331	1470	42829
1	Direct	128141.06	568	456	539	16132
2	Organic Search	93376.88	991	905	956	28247
3	Paid Search	14028.37	164	153	160	4277
4	Display	4728.45	34	34	34	1044

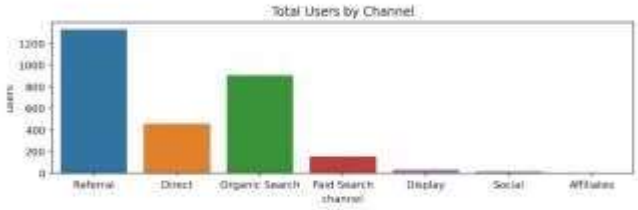
[ ]:
```
fig, (ax1, ax2, ax3, ax4, ax5) = plt.subplots(5, figsize=(10,24))
fig.subplots_adjust(hspace=1)

sns.barplot(x='channel',y='revenue',data=df3,estimator=sum,ax=ax1)

sns.barplot(x='channel',y='transactions',data=df3,estimator=sum,ax=ax2)

sns.barplot(x='channel',y='users',data=df3,estimator=sum,ax=ax3)

sns.barplot(x='channel',y='sessions',data=df3,estimator=sum,ax=ax4)

sns.barplot(x='channel',y='pageviews',data=df3,estimator=sum,ax=ax5)

ax1.set_title('Total Revenue by Channel')
```

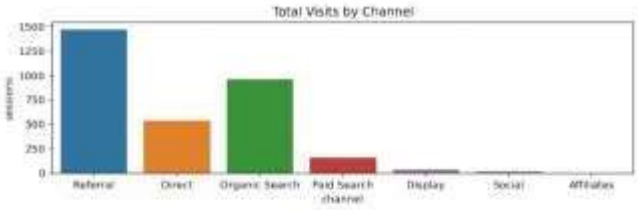
```
ax2.set_title('Total Transactions by Channel')
ax3.set_title('Total Users by Channel')
ax4.set_title('Total Visits by Channel')
ax5.set_title('Total Pageviews by Channel')
```

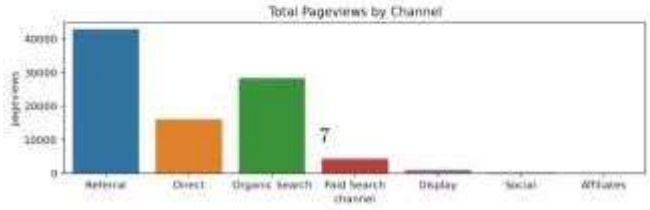
[ ]: `Text(0.5, 1.0, 'Total Pageviews by Channel')`


Total Revenue by Channel
revenue
150000
100000
50000
0
Referral
Direct
Organic Search
Paid Search
channel
Display
Social
Affiliates


Total Transactions by Channel
transactions
1500
1250
1000
750
500
250
0
Referral
Direct
Organic Search
Paid Search
channel
Display
Social
Affiliates


Total Users by Channel
users
1200
1000
800
600
400
200
0
Referral
Direct
Organic Search
Paid Search
channel
Display
Social
Affiliates


Total Visits by Channel
sessions
1500
1250
1000
750
500
250
0
Referral
Direct
Organic Search
Paid Search
channel
Display
Social
Affiliates


Total Pageviews by Channel
pageviews
40000
30000
20000
10000
0
Referral
Direct
Organic Search
Paid Search
channel
Display
Social
Affiliates

2 Behavioural Analysis

[ ]:
```
query = """
    SELECT
        fullVisitorId AS userID,
        AVG(totals.timeOnSite) As avgTimeOnSite,
        SUM(totals.pageviews) AS pageviews,
        SUM(totals.transactions) AS transactions
    FROM
        `bigquery-public-data.google_analytics_sample.ga_sessions_*`
    WHERE
        _TABLE_SUFFIX BETWEEN '20161001' AND '20161230'
    GROUP BY
        1
    ORDER BY 1

"""
safe_query_job = client.query(query)
df4 = safe_query_job.to_dataframe()

df4['transactions'].fillna(0, inplace=True)
df4.head()
```

[ ]:

	userID	avgTimeOnSite	pageviews	transactions
0	0000010278554503158	194.0	8.0	0.0
1	0000020424342248747	297.0	13.0	0.0
2	0000152474579038632	162.0	2.0	0.0
3	0000174067426171406	559.5	30.0	0.0
4	00003347105961862 1	NaN	1.0	0.0

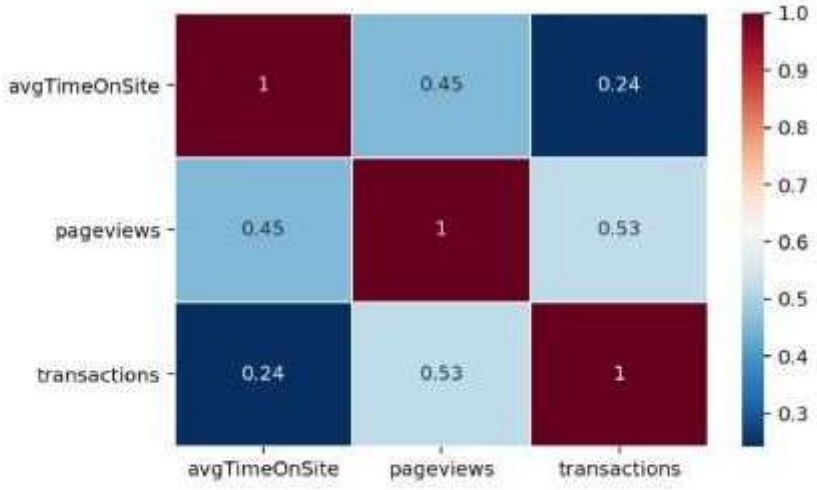
[ ]:
```
pearson_corr = df4.corr(method='pearson')
pearson_corr
```

[ ]:

	avgTimeOnSite	pageviews	transactions
avgTimeOnSite	1.000000	0.454428	0.242365
pageviews	0.454428	1.000000	0.525083
transactions	0.242365	0.525083	1.000000

[ ]:
```
sns.heatmap(pearson_corr,xticklabels=pearson_corr.columns,
            yticklabels=pearson_corr.columns,cmap='RdBu_r',
            annot=True,linewidth=0.5)
```

[ ]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x71fc0377b410>
```



2.0.1 Most visited pages on site ?

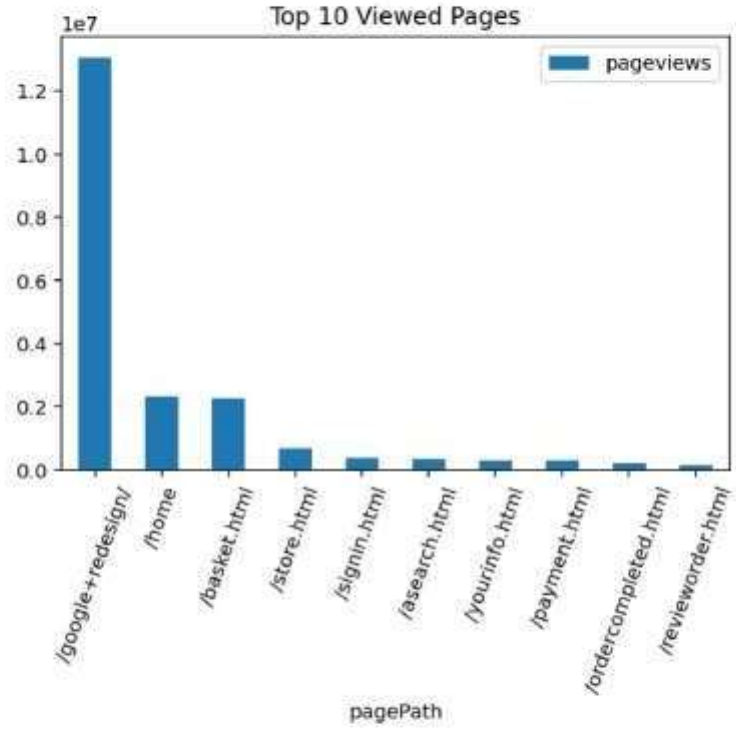
[ ]:
```
query = """
    SELECT
        hits.page.pagePathLevel1 AS pagePath,
        SUM(totals.pageviews) AS pageviews
    FROM
      `bigquery-public-data.google_analytics_sample.ga_sessions_*`,
      UNNEST(hits) AS hits
    WHERE
        _TABLE_SUFFIX BETWEEN '20161001' AND '20161230'
    GROUP BY 1
    ORDER BY 2 DESC
    LIMIT 10
"""
safe_query_job = client.query(query)
df5 = safe_query_job.to_dataframe()
df5
```

[ ]:

	pagePath	pageviews
0	/google+redesign/	13059899
1	/home	2320076
2	/basket.html	2251571
3	/store.html	693064
4	/signin.html	373752

5	/asearch.html	342792
6	/yourinfo.html	309919
7	/payment.html	305266
8	/ordercompleted.html	214672
9	/revieworder.html	144310

[ ]:
```
df5.plot.bar(x='pagePath', y='pageviews', rot=70, title='Top 10 Viewed Pages')
plt.show()
```



[ ]:
```
query = """
    SELECT
        product.v2ProductCategory AS product_category,
        product.v2ProductName AS product_name,
        product.productSKU AS product_sku,
        product.productPrice/1e6 AS product_price,
        product.productQuantity AS product_quantity,
```

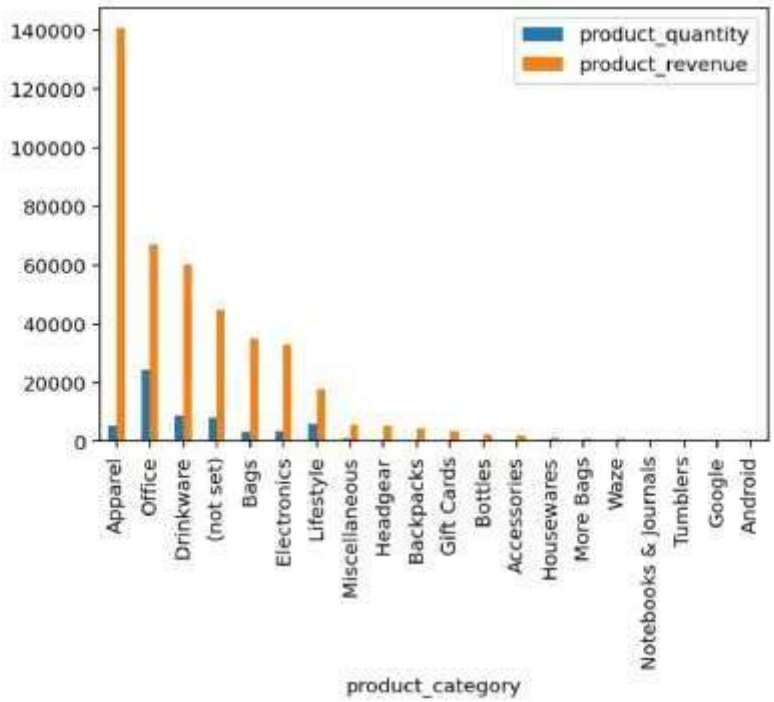
```
        product.productRevenue/1e6 AS product_revenue,
        totals.totalTransactionRevenue/1e6 AS total_revenue
    FROM
      `bigquery-public-data.google_analytics_sample.ga_sessions_*`,
      UNNEST(hits) AS hits,
      UNNEST(hits.product) AS product
    WHERE
        _TABLE_SUFFIX BETWEEN '20161001' AND '20161230'
        AND productRevenue IS NOT NULL
"""
safe_query_job = client.query(query)
df6 = safe_query_job.to_dataframe()
df6.head()
```

[ ]:

	product_category	product_name	product_sku	product_price	product_quantity	product_revenue	total_revenue
0	Headgear	YouTube Trucker Hat	GGOEYHPA003510	21.99	1	23.99	33.98
1	Office	YouTube Notebook and Pen Set	GGOEYOLR080599	7.99	1	9.99	33.98
2	Apparel	Android BTTF Cosmos Graphic Tee	GGOEAAAB034814	15.99	2	35.98	35.98
3	Lifestyle	Straw Beach Mat	GGOEGAYR023499	9.99	2	28.98	28.98
4	Notebooks & Journals	YouTube Spiral Journal with Pen	GGOEYOCR078099	9.99	2	28.98	28.98

[ ]:
```
df6['product_category'].unique()
df6 = df6.replace(['${productItem.product.origCatName}'], 'Miscellaneous')

pivot = pd.pivot_table(df6, index=['product_category'],␣
 ↪values=['product_revenue', 'product_quantity'], aggfunc=np.sum).
 ↪sort_values(by='product_revenue', ascending=False)
pivot.plot(kind='bar')
```

[ ]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x71fc0396f510>
```



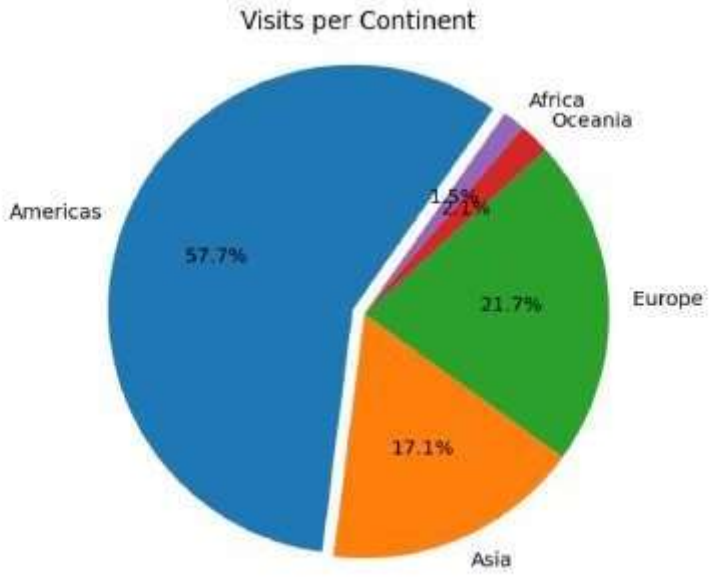
3 Audience Analysis

3.0.1 Visits per Continent

[ ]:
```
query = """
    SELECT geoNetwork AS place
     FROM `bigquery-public-data.google_analytics_sample.ga_sessions_20170801`
"""

safe_config = bigquery.QueryJobConfig(maximum_bytes_billed=10**10)
q_job = client.query(query, job_config=safe_config)

df8 = q_job.to_dataframe()

continent_lst = []
for x in df8['place']:
    x = x['continent']
```

```
    continent_lst.append(x)
df8['continents'] = continent_lst

continent_count = {'Americas': 0, 'Asia': 0, 'Europe': 0, 'Oceania': 0,␣
 ↪'Africa': 0}
for continent in df8['continents']:
    if continent == 'Americas':
        continent_count['Americas'] += 1
    if continent == 'Asia':
        continent_count['Asia'] += 1
    if continent == 'Europe':
        continent_count['Europe'] += 1
    if continent == 'Oceania':
        continent_count['Oceania'] += 1
    if continent == 'Africa':
        continent_count['Africa'] += 1
```

[ ]:
```
labels = continent_count.keys()
data = continent_count.values()
explode = (0.05, 0, 0,0,0)

plt.figure(figsize=(5,5))
plt.pie(data, labels=labels, explode=explode, autopct='%1.1f%%', startangle=65)
plt.title('Visits per Continent')
plt.axis('equal')

plt.show()
```



[ ]:
```
query = """
    SELECT trafficSource.source AS source, COUNT(trafficSource) as counts
    FROM `bigquery-public-data.google_analytics_sample.ga_sessions_20170801`
    GROUP BY source
    HAVING counts >= 10
    ORDER BY counts DESC
"""

safe_config = bigquery.QueryJobConfig(maximum_bytes_billed=10**10)
query_job = client.query(query, job_config=safe_config)

df8 = query_job.to_dataframe()

print(df8)
```

	source	counts
0	(direct)	2166
1	youtube.com	180
2	analytics.google.com	57
3	Partners	52
4	dfa	15
5	google.com	12

[ ]:
```
category_names = df8['source']
results = {'counts': df8['counts']}

def survey(results, category_names):
    labels = list(results.keys())
    data = np.array(list(results.values()))
    data_cum = data.cumsum(axis=1)
    category_colors = plt.get_cmap('RdYlGn')(
        np.linspace(0.15, 0.85, data.shape[1]))

    fig, ax = plt.subplots(figsize=(13, 2.5))
    ax.invert_yaxis()
    ax.xaxis.set_visible(False)
    ax.set_xlim(0, np.sum(data, axis=1).max())

    for i, (colname, color) in enumerate(zip(category_names, category_colors)):
        widths = data[:, i]
        starts = data_cum[:, i] - widths
        ax.barh(labels, widths, left=starts, height=0.5,
                label=colname, color=color)
        xcenters = starts + widths / 2

    ax.legend(ncol=len(category_names), bbox_to_anchor=(0, 1),
              loc='lower left', fontsize='large')

    return fig, ax


survey(results, category_names)
plt.show()
```

[ ]:
```
from time import time

def show_amount_of_data_scanned(query):
    dry_run_config = bigquery.QueryJobConfig(dry_run=True)
    query_job = client.query(query, job_config=dry_run_config)
```

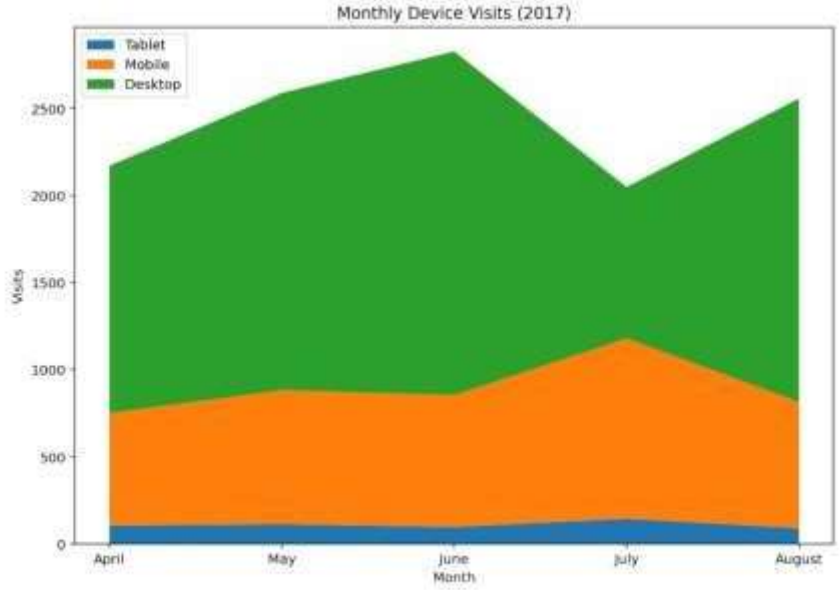
```
    print('Data processed: {} GB'.format(round(query_job.total_bytes_processed /
 ↪ 10**9, 3)))

def show_time_to_run(query):
    time_config = bigquery.QueryJobConfig(use_query_cache=False)
    start = time()
    query_result = client.query(query, job_config=time_config).result()
    end = time()
    print('Time to run: {} seconds'.format(round(end-start, 3)))
```

[ ]:
```
query = """
    SELECT date, august.device.deviceCategory AS device, SUM(august.totals.
 ↪visits) AS visits
    FROM `bigquery-public-data.google_analytics_sample.ga_sessions_20170801`␣
 ↪AS august
    GROUP BY date, device
    UNION ALL
    SELECT date, july.device.deviceCategory as device, SUM(july.totals.visits)␣
 ↪as visits
    FROM `bigquery-public-data.google_analytics_sample.ga_sessions_20170701`␣
 ↪as july
    GROUP BY date, device
    UNION ALL
    SELECT date, june.device.deviceCategory as device, SUM(june.totals.visits)␣
 ↪as visits
    FROM `bigquery-public-data.google_analytics_sample.ga_sessions_20170601`␣
 ↪as june
    GROUP BY date, device
    UNION ALL
    SELECT date, may.device.deviceCategory as device, SUM(may.totals.visits)␣
 ↪as visits
    FROM `bigquery-public-data.google_analytics_sample.ga_sessions_20170501`␣
 ↪as may
    GROUP BY date, device
    UNION ALL
    SELECT date, april.device.deviceCategory as device, SUM(april.totals.
 ↪visits) as visits
    FROM `bigquery-public-data.google_analytics_sample.ga_sessions_20170401`␣
 ↪as april
    GROUP BY date, device
    ORDER BY date, device
    """

safe_config = bigquery.QueryJobConfig(maximum_bytes_billed=10**10)
query_job = client.query(query, job_config=safe_config)
```

```
df9 = query_job.to_dataframe()

print(df9)

show_amount_of_data_scanned(query)
show_time_to_run(query)
```

```
        date   device  visits
0   20170401  desktop    1420
1   20170401   mobile     646
2   20170401   tablet     104
3   20170501  desktop    1706
4   20170501   mobile     770
5   20170501   tablet     112
6   20170601  desktop    1972
7   20170601   mobile     758
8   20170601   tablet      96
9   20170701  desktop     869
10  20170701   mobile    1036
11  20170701   tablet     143
12  20170801  desktop    1742
13  20170801   mobile     725
14  20170801   tablet      89
Data processed: 0.0 GB
Time to run: 5.32 seconds
```

[ ]:
```
desktop = df9['visits'][0::3]
mobile = df9['visits'][1::3]
tablet = df9['visits'][2::3]

x = ['April', 'May', 'June', 'July', 'August']

y = np.vstack([tablet, mobile, desktop])

labels = ['Tablet', 'Mobile', 'Desktop']

plt.figure(figsize=(10,7))
plt.stackplot(x, tablet, mobile, desktop, labels=labels)
plt.title('Monthly Device Visits (2017)')
plt.ylabel('Visits')
plt.xlabel('Month')
plt.legend(loc='upper left')
plt.show()
```



4 Conversion Analysis

[ ]:
```
query = """
    SELECT
        hits.eCommerceAction.action_type AS actions,
        COUNT(fullVisitorId) AS total_hits
    FROM
      `bigquery-public-data.google_analytics_sample.ga_sessions_*`,
      UNNEST(hits) AS hits,
      UNNEST(hits.product) AS product
    WHERE
        _TABLE_SUFFIX BETWEEN '20161001' AND '20161230'
    AND
        (hits.ecommerceaction.action_type != '0' AND hits.ecommerceaction.
  action_type != '4' AND hits.ecommerceaction.action_type != '3')
    GROUP BY
        1
    ORDER BY 1
"""
safe_query_job = client.query(query)
df7 = safe_query_job.to_dataframe()
```

```
df7.head()
```

[ ]:

	actions	total_hits
0	1	111457
1	2	92030
2	5	60395
3	6	19802

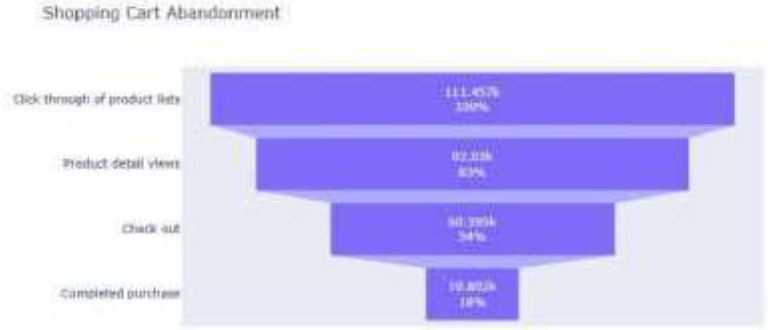
[ ]:
```
query = """
        SELECT
            CASE WHEN hits.eCommerceAction.action_type = '1' THEN 'Click␣
 ↪through of product lists'
                 WHEN hits.eCommerceAction.action_type = '2' THEN 'Product␣
 ↪detail views'
                 WHEN hits.eCommerceAction.action_type = '5' THEN 'Check out'
                 WHEN hits.eCommerceAction.action_type = '6' THEN 'Completed␣
 ↪purchase'
            END AS action,
            COUNT(fullVisitorID) AS users,
        FROM
            `bigquery-public-data.google_analytics_sample.ga_sessions_*`,
                UNNEST(hits) AS hits,
                UNNEST(hits.product) AS product
        WHERE
            _TABLE_SUFFIX BETWEEN '20160801' AND '20170801'
            AND
            (
            hits.eCommerceAction.action_type != '0'
            AND
            hits.eCommerceAction.action_type != '3'
            AND
            hits.eCommerceAction.action_type != '4'
            )
        GROUP BY action
        ORDER BY users DESC
        """

result = client.query(query).result().to_dataframe()
result.head(10)
```

[ ]:

	action	users
0	Click through of product lists	445760
1	Product detail views	362607
2	Check out	248936
3	Completed purchase	74457

```
[ ]: df7['actions'] = df7['actions'].astype(str)
     df7['actions'] = df7['actions'].
       replace(['0','1','2','3','4','5','6'],['Unknown','Click through of product
       lists','Product detail views','Add product(s) to cart','Remove products from
       cart',
                                        'Check out','Completed purchase'])
```

```
[ ]: fig = go.Figure(go.Funnel(y = df7['actions'],x = df7['total_hits'],textposition
       = 'inside',textinfo = 'value+percent initial'))
     fig.update_layout(title_text = 'Shopping Cart Abandonment')
     fig.show()
```




```
[ ]: fig = go.Figure(go.Funnel(y = result['action'], x = result['users'],
       textposition = 'inside', textinfo = 'value+percent initial'))
     fig.update_layout(title_text = 'Google Merchandise Store Conversion Path')

     fig.show()
```

[ ]:
```
#Bounce rate based on source
#If you dont understand what is bounce rate or how to calculate it check this␣
 ↪link
#https://support.google.com/analytics/answer/1009409?hl=en
bounce_rate = """
SELECT
total_visits,
source,
 round(( ( total_no_of_bounces / total_visits ) * 100 ),2) AS bounce_rate
FROM (
SELECT
date AS date,
trafficSource.source AS source,
COUNT (fullVisitorId) AS total_visits,
SUM ( totals.bounces ) AS total_no_of_bounces
FROM `bigquery-public-data.google_analytics_sample.ga_sessions_*`
WHERE _table_suffix BETWEEN '20170701'
AND '20170730'
GROUP BY
source,
date)
ORDER BY
--ordering by total visits we can get sources who bringed the most traffic into␣
 ↪the site.
date,
total_visits DESC
LIMIT 10
```

```
"""
safe_query_job= client.query(bounce_rate)
bounce_rate_df= safe_query_job.to_dataframe()
bounce_rate_df
```

[ ]:

	total_visits	source	bounce_rate
0	1172	google	56.57
1	335	(direct)	54.03
2	255	m.facebook.com	59.61
3	162	youtube.com	74.69
4	46	facebook.com	47.83
5	19	analytics.google.com	68.42
6	17	Partners	64.71
7	8	baidu	75.00
8	7	google.com	28.57
9	5	quora.com	100.00

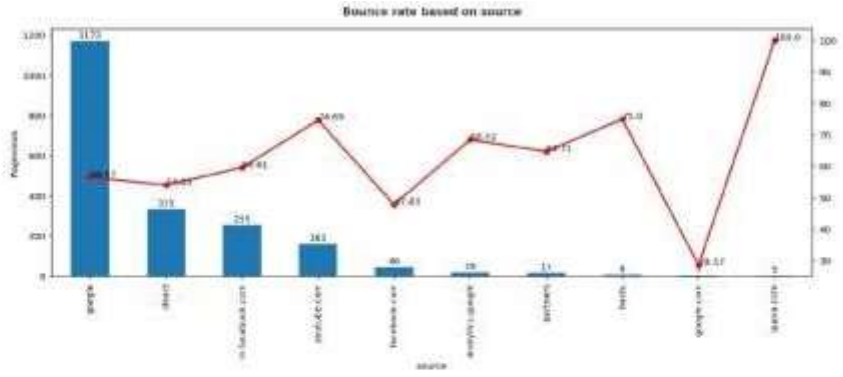
[ ]:
```
bounce_pageviews_query = """
WITH bounce_pageviews AS
                    (SELECT
                    COUNT(fullvisitorId)AS pageviews,
                    SUM(totals.bounces) AS bounces,
                    hits.page.pagePath AS page
                    FROM `bigquery-public-data.google_analytics_sample.
  ga_sessions_*`
                   ,UNNEST(hits) as hits
                   WHERE  _table_suffix BETWEEN '20170701'
                   AND '20170730'
                   AND hits.type = 'PAGE'
                   GROUP BY page                          )

    SELECT
    pageviews,
    page,
    IFNULL(((bounces / pageviews)*100),0) AS bounce_rate
    FROM bounce_pageviews
    GROUP BY pageviews, page, bounces
    ORDER BY PAGEVIEWS DESC
    LIMIT 20
    """
safe_query_job= client.query(bounce_pageviews_query)
bounce_df= safe_query_job.to_dataframe()
bounce_df
```

[ ]:

	pageviews	page	bounce_rate
0	59491	/home	25.879545

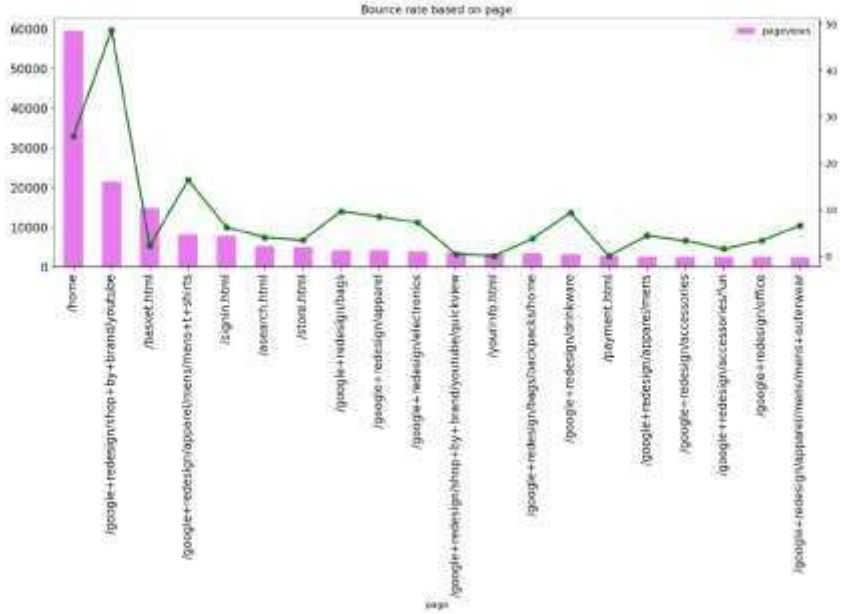
1	21383	/google+redesign/shop+by+brand/youtube	48.664827
2	14864	/basket.html	2.186491
3	8193	/google+redesign/apparel/mens/mens+t+shirts	16.404248
4	7822	/signin.html	6.213245
5	5355	/asearch.html	4.033613
6	5091	/store.html	3.417796
7	4309	/google+redesign/bags	9.584590
8	4301	/google+redesign/apparel	8.463148
9	4118	/google+redesign/electronics	7.236523
10	3717	/google+redesign/shop+by+brand/youtube/quickview	0.403551
11	3481	/yourinfo.html	0.000000
12	3469	/google+redesign/bags/backpacks/home	3.718651
13	3235	/google+redesign/drinkware	9.304482
14	2884	/payment.html	0.000000
15	2608	/google+redesign/apparel/mens	4.371166
16	2575	/google+redesign/accessories	3.300971
17	2560	/google+redesign/accessories/fun	1.523438
18	2518	/google+redesign/office	3.296267
19	2463	/google+redesign/apparel/mens/mens+outerwear	6.577345

[ ]:
```
n=bounce_rate_df["total_visits"]
plt.figure (figsize=(15,5))
ax = bounce_rate_df["total_visits"].plot(kind='bar', use_index=True, )
ax2 = ax.twinx()
ax2.plot(bounce_rate_df["bounce_rate"].values, linestyle='-', marker='o',␣
 ↪linewidth=2.0,color="red")
ax.set_xticklabels(('google', 'direct', 'm.facebook.com', 'youtube.com',␣
 ↪'facebook.com', 'analytics.google', 'partners', 'baidu', 'google.com',␣
 ↪'quora.com'))
ax.set_xlabel('source')
ax.set_ylabel("Pageviews")
ax.set_title('Bounce rate based on source', pad=15, color='#333333',
             weight='bold')
#Making labels for bar chart
rects= ax.patches
labels = bounce_rate_df["total_visits"]

for rect,label in zip(rects,labels):
    height = rect.get_height()
    ax.text(rect.get_x() + rect.get_width()/ 2, height + 5, label,
            ha='center', va='bottom')
#Making labels for line chart
for i, j in bounce_rate_df.bounce_rate.items():
    ax2.annotate(str(j), xy=(i,j))
```



[ ]:
```
ax= bounce_df.plot.bar(x='page',
                       y='pageviews',
                       figsize=(15,5),
                       color='#EE75F3',
                       fontsize=13,
                       title= 'Bounce rate based on page'
                      )
ax2=ax.twinx()
ax2.plot(bounce_df["bounce_rate"].values, linestyle ='-', marker='o',␣
 ↪linewidth=2.0,color="green")
```

[ ]:
```
[<matplotlib.lines.Line2D at 0x71fbf46ce510>]
```




```
[ ]: #RETURNING VISITORS(%)(RVR)
returning_visitor="""
SELECT
date,
total_visitors,
returning_visitors,
--All about RVR https://contently.com/2015/08/18/
  how-loyal-are-your-customers-this-metric-has-the-answer/
((SUM(returning_visitors)/SUM(total_visitors))*100) AS RVR
FROM(SELECT
date,
COUNT(CASE WHEN visitNumber > 1 THEN fullvisitorId ELSE NULL END) AS␣
  ↪returning_visitors,
COUNT(fullvisitorId) AS total_visitors
FROM `bigquery-public-data.google_analytics_sample.ga_sessions_*`
WHERE _TABLE_SUFFIX BETWEEN '20170701' AND '20170731'
GROUP BY date
ORDER BY date)
GROUP BY date, total_visitors, returning_visitors
ORDER BY date
"""
```

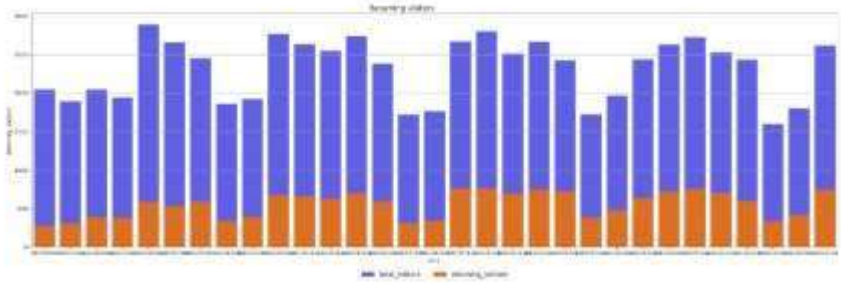
```
safe_query_job= client.query(returning_visitor)
returning_visitors_df= safe_query_job.to_dataframe()
returning_visitors_df
```

[ ]:

	date	total_visitors	returning_visitors	RVR
0	20170701	2048	275	13.427734
1	20170702	1895	311	16.411609
2	20170703	2046	386	18.866080
3	20170704	1938	374	19.298246
4	20170705	2885	589	20.415945
5	20170706	2658	533	20.052671
6	20170707	2450	591	24.122449
7	20170708	1859	333	17.912856
8	20170709	1921	388	20.197814
9	20170710	2769	679	24.521488
10	20170711	2635	658	24.971537
11	20170712	2554	625	24.471417
12	20170713	2741	698	25.465159
13	20170714	2382	592	24.853065
14	20170715	1721	314	18.245206
15	20170716	1766	341	19.309173
16	20170717	2671	760	28.453763
17	20170718	2804	758	27.032810
18	20170719	2514	691	27.486078
19	20170720	2668	744	27.886057
20	20170721	2427	725	29.872270
21	20170722	1724	382	22.157773
22	20170723	1966	472	24.008138
23	20170724	2436	638	26.190476
24	20170725	2631	720	27.366021
25	20170726	2725	756	27.743119
26	20170727	2529	703	27.797548
27	20170728	2433	600	24.660912
28	20170729	1597	332	20.788979
29	20170730	1799	417	23.179544
30	20170731	2620	746	28.473282

[ ]:
```
%matplotlib inline
fig = plt.figure(figsize=(27,8))
ax = fig.add_subplot(111)
# Variables that define subsets of the data, which will be drawn on separate␣
  ↪facets in the grid.
sns.barplot(x=returning_visitors_df["date"],
            y=returning_visitors_df["total_visitors"]
            ,data=returning_visitors_df,label = 'total_visitors'
            ,color='#4848FF'
            ,ax=ax)
```

```
#Dont know how to explain better but here I'm making barchart on another
 barchart. Also known as stacked barchart.
sns.barplot(x=returning_visitors_df["date"]
            ,y=returning_visitors_df["returning_visitors"]
            ,data=returning_visitors_df,label = 'returning_visitors'
            ,color='#FF6800',ax=ax)
plt.title('Returning visitors',fontsize=14,ha='right')
# Put a legend below current axis
plt.legend(loc='upper center', bbox_to_anchor=(0.5, -0.08),fancybox=True,
 shadow=False, ncol=5,fontsize=12)
plt.grid(axis='y')
```



CONCLUSION:

Customer Analytics helps businesses draw useful insights, create more useful products, deliver better services, and develop more profitable business decisions. Using data mining approaches, big data queries and visualization libraries, patterns can be identified in customer behaviors, and its correlation to sitewide performance. These analytics directly led to quantitative analysis of frequency of purchases, customer retentions, bounce back rates and customer conversion rates at various points of the purchasing process.