HUMBER INSTITUTE OF TECHNOLOGY

AND ADVANCED LEARNING

(HUMBER COLLEGE)

ASSIGNMENT NAME: In Class Group Assignment

Group No – 7

Sr No.	Name	Student Number
1	Vidhya Shanker Goel	N01650917
2	Samyuktha Lakshmi Narayanan	N01654489
3	Mohammad Haris Rasheed	N01609771
4	Himalay Dwivedi	N01647742
5	Nirupama Perumal Perumalthevar	N01650845
6	Rohith Prabhakaran Latha	N01653327

STEP 1: LOAD THE DATATSET IN DATABRICKS NOTEBOOK AND DISPLAY FIRST 10 ROWS.

```
df = spark.read.option("sep", ",") \
    .option("header", True) \
    .option("inferSchema", True) \
    .csv("s3n://humber-lfb-databricks-class-files/bigdata1_ga.csv")

# Show the first 10 rows of the DataFrame to inspect the data
display(df.limit(10))
```

Table

	month	format	device_type	bid_type	network_id
1	Nov-2019	display	desktop	cpc	20
2	Nov-2019	display	desktop	cpc	75
3	Nov-2019	display	desktop	cpc	85
4	Nov-2019	display	desktop	cpc	86
5	Nov-2019	display	desktop	cpc	87
6	Nov-2019	display	desktop	cpc	94
7	Nov-2019	display	desktop	cpc	95
8	Nov-2019	display	desktop	cpc	106
9	Nov-2019	display	desktop	cpc	109
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

10 rows

STEP 2: PRELIMINARY ANALYSIS TO UNDERSTAND THE DATA DISTRIBUTION, MISSING VALUES AND UNIQUE VALUES.

- **Code for Data Distribution and Missing Values**

```
# Display summary statistics for numerical columns
df.describe().display()

# Check for missing values in each column
df.select([count(when(col(c).isNull(), c)).alias(c) for c in df.columns]).display()
```

- **Output**

Table

	summary	month	format	device_type	bid_type
1	count	[illegible]	[illegible]	[illegible]	[illegible]
2	mean	null	null	null	null
3	stddev	null	null	null	null
4	min	Apr-2020	Audio	connected-device	cpc
5	max	Sep-2020	video	unknown	cpm

5 rows

Table

	month	format	device_type	bid_type	network_id
1	0	0	0	0	0

- **Code for Unique values**

```
# Count unique values in categorical columns
categorical_columns = ["month", "format", "device_type", "bid_type", "network_id"]
for column in categorical_columns:
    print(f"Unique values in {column}: {df.select(column).distinct().count()}")
```

- **Output**

```
Unique values in month: 12
Unique values in format: 5
Unique values in device_type: 9
Unique values in bid_type: 4
Unique values in network_id: 93
```

STEP 3: PERFORMANCE METRICS

a) Calculate the Click-Through Rate (CTR) for each month.

- **Code and Output**

```
df_ctr = df.withColumn("CTR", col("clicks") / col("impressions")) \
            .groupBy("month") \
            .agg(sum("CTR").alias("Total_CTR"))

display(df_ctr)
```

Table

	month	Total_CTR
1	Jun-2020	0.24066408206969167
2	Oct-2020	0.5613934226566515
3	Mar-2020	0.7568150549324686
4	May-2020	0.4819151542658173
5	Dec-2019	0.860034171230304
6	Sep-2020	1.1882847572023085
7	Jan-2020	0.46624284251332204
8	Aug-2020	0.3284417613768979
9	Jul-2020	0.6083789196310952
10	Feb-2020	0.351244935496737
11	Apr-2020	1.856179703887004
12	Nov-2019	2.3738953935803027

12 rows

b) Calculate the Cost Per Click (CPC) and Cost Per Mille (CPM)

- **Code and Output**

```
# Calculate CPC
df = df.withColumn("CPC", col("spend") / col("clicks"))

# Calculate CPM
df = df.withColumn("CPM", (col("spend") / col("impressions")) * 1000)

display(df.select("month", "CPC", "CPM"))
```

Table

	month	1.2 CPC	1.2 CPM
1	Nov-2019	1.50122448979591...	null
2	Nov-2019	1.35357142857142...	null
3	Nov-2019	1.47995555555555...	null
4	Nov-2019	1.23133333333333...	null
5	Nov-2019	1.36695067264573...	null
6	Nov-2019	1.28794117647058...	null
7	Nov-2019	1.28395209580838...	null
8	Nov-2019	1.54232558139534...	null
9	Nov-2019	1.673	null
10	Nov-2019	1.30285714285714...	null
11	Nov-2019	1.29405940594059...	null
12	Nov-2019	1.42125	null
13	Nov-2019	1.2525	null
14	Nov-2019	1.49857142857142...	null
15	Nov-2019	1.34798076923076...	null

10,000+ rows | Truncated data due to row limit

STEP 4: VISUALIZATION OF THE DATASET

a) Plot a time series graph to show monthly spend.

- Code and Output

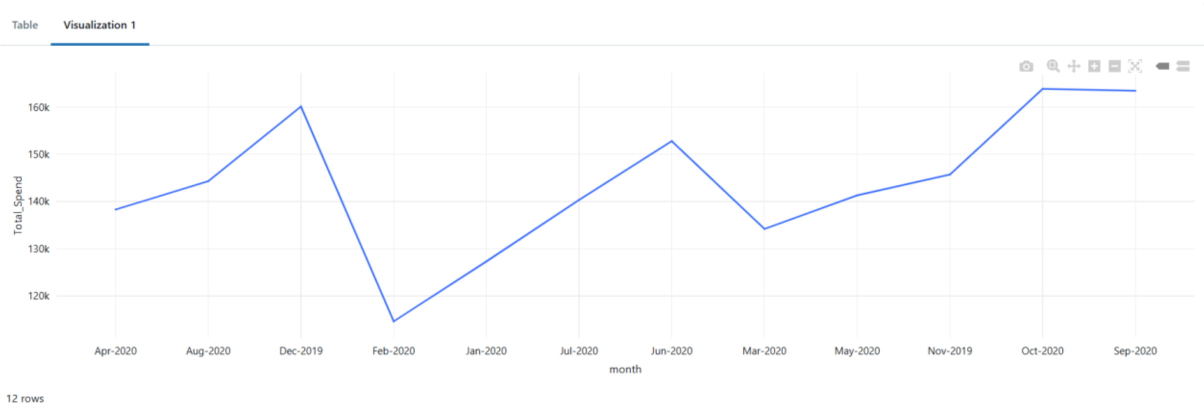
```
# Aggregate spend by month
monthly_spend = df.groupBy("month").agg(sum("spend").alias("Total_Spend"))

display(monthly_spend)
```

Table   Visualization 1

	month	Total_Spend
1	Jun-2020	152796.65631299984
2	Oct-2020	163849.856116964
3	Mar-2020	134197.77977173286
4	May-2020	141298.22913756393
5	Dec-2019	160112.8264998041
6	Sep-2020	163455.27900642302
7	Jan-2020	127322.76243036397
8	Aug-2020	144304.93155699
9	Jul-2020	140340.16237462606
10	Feb-2020	114626.18083690015
11	Apr-2020	138305.90126029693
12	Nov-2019	145717.09502910904

12 rows

- Resulting Visualization based on the Output



b) Visualize the distribution of impressions and clicks across different device types

- Code and Output

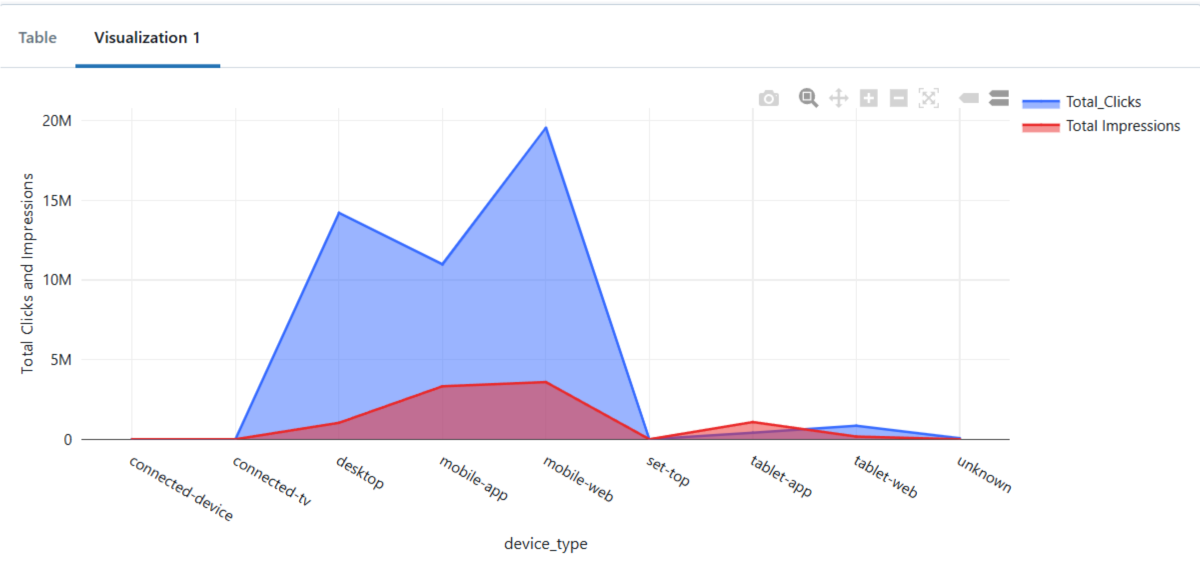
```
# Aggregate impressions and clicks by device type
device_metrics = df.groupBy("device_type").agg(sum("impressions").alias("Total_Impressions"),
                                               sum("clicks").alias("Total_Clicks"))

display(device_metrics)
```

Table   Visualization 1

	device_type	Total_Impressions	Total_Clicks
1	desktop	14211172	1031103
2	unknown	60984	167
3	set-top	4164	0
4	tablet-app	412716	1078939
5	mobile-app	10981783	3325041
6	tablet-web	847287	168155
7	connected-device	2171	0
8	connected-tv	4900	0
9	mobile-web	19558743	3589196

9 rows

- Resulting Visualization based on the Output



c) Show a bar chart of engagements across different ad formats.

- Code and Output

```
# Aggregate engagements by ad format
ad_format_engagements = df.groupBy("format").agg(sum("engagements").alias("Total_Engagements"))

display(ad_format_engagements)
```

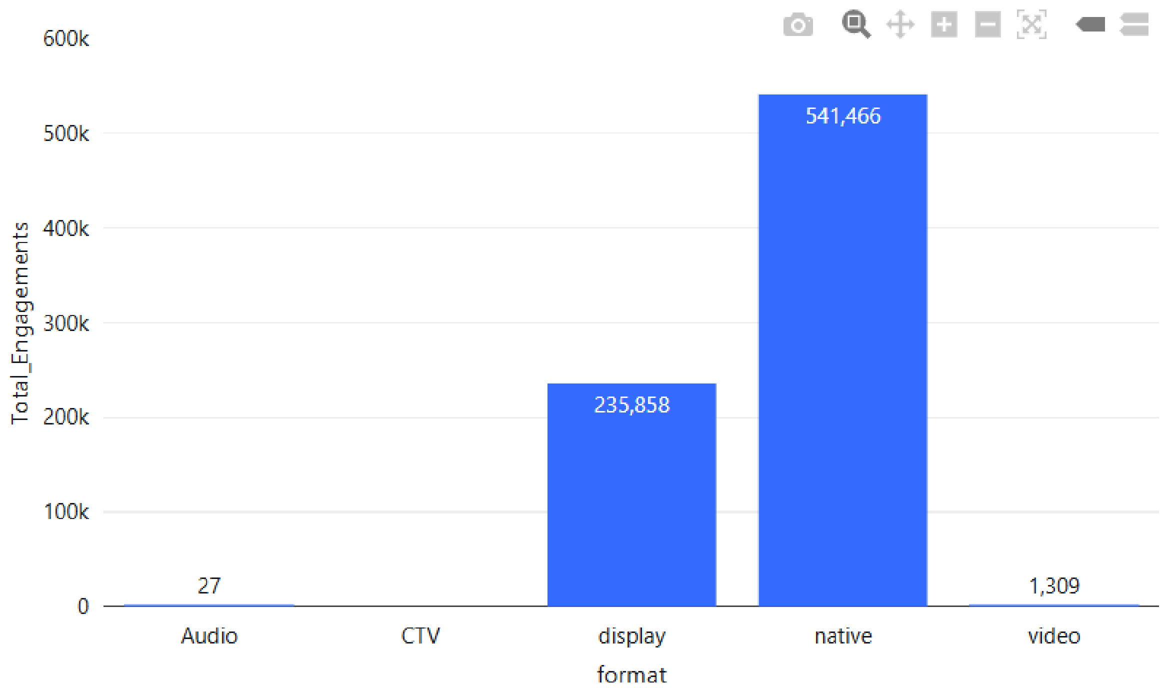
Table | Visualization 1

	format	Total_Engagements
1	display	235858
2	CTV	0
3	video	1309
4	Audio	27
5	native	541466

5 rows

- **Resulting Visualization based on the Output**

Table | Visualization 1



5 rows

STEP 5: ENGAGEMENT ANALYSIS

a) Analyse the correlation between viewable impressions and engagements.

- Code and Output

```
# Calculate correlation
viewable_corr = df.stat.corr("viewable_imps", "engagements")
print(f"Correlation between Viewable Impressions and Engagements: {viewable_corr}")
```

```
Correlation between Viewable Impressions and Engagements: 0.4375628324188831
```

b) Investigate the ratio of video start to video complete. Which month had the highest drop-offs?

- Code and Output

```
# Calculate drop-off ratio
video_ratio = df.withColumn("Drop_Offs", (col("video_start") - col("video_complete")) / col("video_start"))

# Find the month with the highest drop-offs
highest_drop_offs = video_ratio.groupBy("month") \
                               .agg(sum("Drop_Offs").alias("Total_Drop_Offs")) \
                               .orderBy(col("Total_Drop_Offs").desc())

display(highest_drop_offs)
```

Table

	month	Total_Drop_Offs
1	Oct-2020	72.64194314724473
2	Jul-2020	68.78931478165
3	Aug-2020	67.88625169935722
4	Sep-2020	64.51052969098535
5	Dec-2019	63.81469340909828
6	Nov-2019	59.75284657096219
7	Jan-2020	58.93456874093329
8	Feb-2020	58.87021519491784
9	Mar-2020	57.44223066265956
10	Apr-2020	56.683825522479744
11	Jun-2020	55.622783641355305
12	May-2020	53.52432659616489

12 rows

STEP 6: TWO INSIGHTS ON THE ANALYSIS

First:-

Observation: The number of impressions for certain device types, such as tablet-app and mobile-web, is significantly high, but the corresponding number of clicks doesn't proportionally match the impression count. Insight: This suggests that while ads are being displayed frequently on these devices, they may not be as engaging or relevant to users, leading to lower click-through rates. This could be an opportunity to optimize ad content or targeting strategies for these specific devices.

Second:-

Observation: The set-top and connected-tv device types show zero clicks, despite having impressions. Insight: This could indicate a mismatch between the content of the ads and the context in which they are displayed on these devices. These device types might require a different approach, such as more interactive or video-based content, to engage users effectively.