Analyzing Marketing Campaigns

—

Importing Liberaries

In [460…

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.ticker as mticker
import matplotlib.patches as mpatches
import matplotlib.lines as mlines
import matplotlib.patheffects as path_effects
import seaborn as sns
```

The Dataset

In [461…

```
#Loading The Marketing Dataset
marketing=pd.read_csv('marketing.csv')
marketing.head()
```

Out[461…

	user_id	date_served	marketing_channel	variant	converted	language_displa
0	a100000029	1/1/18	House Ads	personalization	True	Eng
1	a100000030	1/1/18	House Ads	personalization	True	Eng
2	a100000031	1/1/18	House Ads	personalization	True	Eng
3	a100000032	1/1/18	House Ads	personalization	True	Eng
4	a100000033	1/1/18	House Ads	personalization	True	Eng

Data Assessing

1 − Data Types and Null Values :

In [462…

```
# Examining data types & null values:
marketing.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10037 entries, 0 to 10036
Data columns (total 12 columns):
 #   Column               Non-Null Count  Dtype 
---  ------               --------------  ----- 
 0   user_id              10037 non-null  object
 1   date_served          10021 non-null  object
 2   marketing_channel    10022 non-null  object
 3   variant              10037 non-null  object
 4   converted            10022 non-null  object
 5   language_displayed   10037 non-null  object
 6   language_preferred   10037 non-null  object
 7   age_group            10037 non-null  object
 8   date_subscribed      1856 non-null   object
 9   date_canceled        577 non-null    object
 10  subscribing_channel  1856 non-null   object
 11  is_retained          1856 non-null   object
dtypes: object(12)
memory usage: 941.1+ KB
```

2 − Dataset Description :

In [463…

```
# Dataset Description:
print(f'- The Marketing Dataset consists of {marketing.shape[0]} Rows and {marketin

print (f'\n- The dataset consists of {marketing.user_id.nunique()} user.\n')

def dates(df,cols):
    start= df[cols].astype('datetime64[ns]').min().strftime('%Y-%m-%d')
    end= df[cols].astype('datetime64[ns]').max().strftime('%Y-%m-%d')
    return f'''
    Start: {start}
    End  : {end}\n'''
print ('- The Data Selection:',dates(marketing,'date_served'))
print ('- The Subscription Dates:',dates(marketing,'date_subscribed'))
print ('- The Subscription Cancellation occured within:',dates(marketing,'date_canc

def col_uniques (df,cols):
    details = []
    for x, y in enumerate(df[cols].unique(),start=1):
        details.append(f"  {x} - {y}")
    return "\n ".join(details)
print('- The Marketing Channels are as follows:\n',col_uniques(marketing,'marketing
print('\n- The Variant categories are as follows:\n',col_uniques(marketing,'variant
print('\n- The Converted column is classified into:\n',col_uniques(marketing,'conve
print('\n- The Displayed Languages are as follows:\n',col_uniques(marketing,'langua
print('\n- The Preferred Languages are as follows:\n',col_uniques(marketing,'langua
print('\n- The Age Groups are classified as follows:\n',col_uniques(marketing,'age_
```

```
print('\n- The Subscribing Channels are as follows:\n',col_uniques(marketing,'subsc
print('\n- The is_retained column is classified into:\n',col_uniques(marketing,'is_
```

```
- The Marketing Dataset consists of 10037 Rows and 12 Columns

- The dataset consists of 7309 user.

- The Data Selection:
    Start: 2018-01-01
    End  : 2018-01-31

- The Subscription Dates:
    Start: 2018-01-01
    End  : 2018-01-31

- The Subscription Cancellation occured within:
    Start: 2018-01-05
    End  : 2018-05-09

- The Marketing Channels are as follows:
   1 - House Ads
   2 - Push
   3 - Facebook
   4 - Instagram
   5 - Email
   6 - nan

- The Variant categories are as follows:
   1 - personalization
   2 - control

- The Converted column is classified into:
   1 - True
   2 - False
   3 - nan

- The Displayed Languages are as follows:
   1 - English
   2 - German
   3 - Arabic
   4 - Spanish

- The Preferred Languages are as follows:
   1 - English
   2 - German
   3 - Arabic
   4 - Spanish

- The Age Groups are classified as follows:
   1 - 0-18 years
   2 - 19-24 years
   3 - 24-30 years
   4 - 30-36 years
   5 - 36-45 years
   6 - 45-55 years
   7 - 55+ years

- The Subscribing Channels are as follows:
   1 - House Ads
```

```
  2 - Email
  3 - Push
  4 - Facebook
  5 - Instagram
  6 - nan

- The is_retained column is classified into:
  1 - True
  2 - False
  3 - nan
```

3 − Summary Statistics :

In [464…

```
# Summary Statistics
marketing.describe()
```

Out[464…

	user_id	date_served	marketing_channel	variant	converted	language_displaye
count	10037	10021	10022	10037	10022	100:
unique	7309	31	5	2	2	
top	a100000882	1/15/18	House Ads	control	False	Engli:
freq	12	789	4733	5091	8946	97!

4 − Duplicated Values :

In [465…

```
# Create a function to identify Duplicated Values:
def duplicates (df):
    if df.duplicated().sum() == 0:
        result= f'The Dataset has no Duplicated Values with {marketing.shape[0]} Ro
    else:
        result =  f'''
- The Dataset has {df.duplicated().sum()} Duplicated rows and their indexes are as
"{", ".join(map(str,df[df.duplicated()].index.to_list()))}"'''
    return result

# Checking for duplicates:
print(duplicates(marketing))
```

```
- The Dataset has 37 Duplicated rows and their indexes are as follows:

"470, 478, 894, 895, 954, 955, 1004, 1005, 1027, 1047, 1051, 3022, 3166, 3196, 3198,
3310, 3498, 3642, 3801, 3803, 4083, 4124, 4129, 4134, 6880, 7440, 7488, 8452, 8454,
8456, 8458, 8486, 8488, 8500, 8502, 8504, 8506"
```

5 − Missing Values :

In [466…

```
# Create a function to identify the Null Values:
def missing(df):
    if df.isna().sum().sum() == 0:
        result='The Dataset has no NULL Values'
    else:
        result = f'''The Dataset has {df.isna().sum().sum()} NULL Values that are d
    return(result)

# Checking for Missing Values:
print(missing(marketing))
marketing.isna().sum().reset_index().rename(columns={'index':'Column_Name',0:'NULLs
```

```
The Dataset has 34049 NULL Values that are distributed as follows:
```

Out[466…

Column_Name	NULLs_Count
user_id	0
date_served	16
marketing_channel	15
variant	0
converted	15
language_displayed	0
language_preferred	0
age_group	0
date_subscribed	8181
date_canceled	9460
subscribing_channel	8181
is_retained	8181

> *Note* :
> ***date_subscribed, date_canceled, subscribing_channel, & is_retained:***
>
> - *These values are naturally missing depending on whether the user subscribed or not.*
> - *Some exceptions may arise that would require a precautionary measure to make sure that the data values are consistent with each other.*
> - ***For Example,** A handling step to make sure that if a user converted, the subscription information must be addressed as well.*

In [467…

```
# Detemining the indexes of the null values for columns:
def missing_indexs (df,cols):
```

```
    result= df[cols].isna().sum()
    details= ", ".join(map(str,df[df[cols].isna()==True].index.to_list()))

    return f'''
- The "{cols}" Column has {result} NULL Values and their Indexes are as follows:\n
"{details}"\n'''

# 1- date_served
print(missing_indexs(marketing,'date_served'))

# 2- marketing_channel
print(missing_indexs(marketing,'marketing_channel'))

# 3- converted
print(missing_indexs(marketing,'converted'))
```

```
- The "date_served" Column has 16 NULL Values and their Indexes are as follows:

"7038, 9944, 9945, 9946, 9947, 9948, 9949, 9950, 9951, 9952, 9953, 9954, 9955, 9956,
9957, 9958"


- The "marketing_channel" Column has 15 NULL Values and their Indexes are as follow
s:

"9944, 9945, 9946, 9947, 9948, 9949, 9950, 9951, 9952, 9953, 9954, 9955, 9956, 9957,
9958"


- The "converted" Column has 15 NULL Values and their Indexes are as follows:

"9944, 9945, 9946, 9947, 9948, 9949, 9950, 9951, 9952, 9953, 9954, 9955, 9956, 9957,
9958"
```

> *Note*:
> ***date_served, marketing_channel, & converted columns:***
> *The three columns share the same missing rows (except for date_served index 7038)*

5 − User Behavior:

In [468…

```
# Counting users' frequency:
user_freq= marketing.user_id.value_counts().reset_index()
user_freq.columns=['user_id','frequency']
user_freq.sort_values('frequency',ascending=False).head()
```

Out[468…

	user_id	frequency
0	a100000882	12
10	a100000886	10
1	a100000892	10
17	a100000894	10
16	a100000893	10

In [469…

```
# Calculating the number of users based on their frequency
user_freq.groupby('frequency').user_id.count().reset_index()\
        .rename(columns={'frequency':'user_engagement','user_id':'num_users'}).sort
```

Out[469…

	user_engagement	num_users
7	12	1
6	10	17
5	8	2
4	5	13
3	4	62
2	3	126
1	2	2060
0	1	5028

In [470…

```
# Assessing the user_id with 12 engagements:
user_freq.query('frequency==12')
```

Out[470…

	user_id	frequency
0	a100000882	12

In [471…

```
# Assessing the user_id with 12 engagements (user_id: a100000882):
marketing.query('user_id=="a100000882"')
```

Out[471…

	user_id	date_served	marketing_channel	variant	converted	language_disp
874	a100000882	1/14/18	Instagram	personalization	True	E
875	a100000882	1/14/18	Instagram	personalization	True	E
876	a100000882	1/18/18	Instagram	control	True	E
877	a100000882	1/18/18	Instagram	control	True	E
878	a100000882	1/1/18	House Ads	control	False	E
879	a100000882	1/1/18	House Ads	control	False	E
880	a100000882	1/2/18	House Ads	personalization	False	E
881	a100000882	1/2/18	House Ads	personalization	False	E
882	a100000882	1/3/18	House Ads	personalization	False	E
883	a100000882	1/3/18	House Ads	personalization	False	E
884	a100000882	1/2/18	House Ads	control	False	E
885	a100000882	1/2/18	House Ads	control	False	E

In [472…

```
# user_id with 10 engagements:
user_freq.query('frequency==10').head()
```

Out[472…

	user_id	frequency
1	a100000892	10
2	a100000884	10
3	a100000877	10
4	a100000878	10
5	a100000879	10

In [473…

```
# Assessing the a user_id with 10 engagements (user_id: a100000878):
marketing.query('user_id=="a100000878"')
```

Out[473…

	user_id	date_served	marketing_channel	variant	converted	language_disp
834	a100000878	1/10/18	Email	control	True	E
835	a100000878	1/10/18	Email	control	True	E
836	a100000878	1/14/18	Email	control	True	E
837	a100000878	1/14/18	Email	control	True	E
838	a100000878	1/2/18	House Ads	control	False	E
839	a100000878	1/2/18	House Ads	control	False	E
840	a100000878	1/3/18	House Ads	personalization	False	E
841	a100000878	1/3/18	House Ads	personalization	False	E
842	a100000878	1/3/18	House Ads	control	False	E
843	a100000878	1/3/18	House Ads	control	False	E

In [474…

```
# user_id with 8 engagements:
user_freq.query('frequency==8')
```

Out[474…

	user_id	frequency
18	a100000875	8
19	a100000876	8

In [475…

```
# Assessing the a user_id with 8 engagements:
marketing.query('user_id=="a100000875"')
```

Out[475…

	user_id	date_served	marketing_channel	variant	converted	language_disp
808	a100000875	1/7/18	Instagram	personalization	True	E
809	a100000875	1/7/18	Instagram	personalization	True	E
810	a100000875	1/11/18	Instagram	control	True	E
811	a100000875	1/11/18	Instagram	control	True	E
812	a100000875	1/2/18	House Ads	personalization	False	E
813	a100000875	1/2/18	House Ads	personalization	False	E
814	a100000875	1/3/18	House Ads	control	False	E
815	a100000875	1/3/18	House Ads	control	False	E

In [476…

```
# user_id with 5 engagements:
user_freq.query('frequency==5').head()
```

Out[476…

	user_id	frequency
20	a100002370	5
21	a100000858	5
22	a100002368	5
23	a100002369	5
24	a100000857	5

In [477…

```
# Assessing the a user_id with 5 engagements:
marketing.query('user_id=="a100002369"')
```

Out[477…

	user_id	date_served	marketing_channel	variant	converted	language_displayed
4125	a100002369	1/23/18	House Ads	control	False	English
4126	a100002369	1/25/18	Instagram	control	False	English
4127	a100002369	1/20/18	Facebook	control	False	English
4128	a100002369	1/20/18	Instagram	control	False	English
4129	a100002369	1/20/18	Facebook	control	False	English

In [478…

```
# Assessing if any of users (who saw the ad 5 times) converted:
marketing[marketing['user_id'].isin(user_freq.query('frequency == 5').user_id.to_li
```

Out[478…

	user_id	date_served	marketing_channel	variant	converted	language_disp
739	a100000857	1/28/18	Email	control	True	E
744	a100000858	1/29/18	Push	personalization	True	E
749	a100000859	1/30/18	Facebook	personalization	True	E
754	a100000860	1/23/18	Instagram	personalization	True	E
759	a100000861	1/24/18	Instagram	personalization	True	E
764	a100000862	1/25/18	Instagram	personalization	True	E
769	a100000863	1/26/18	Instagram	personalization	True	E
774	a100000864	1/27/18	Instagram	personalization	True	E

> *Notes*:
>
> - ***Users may be exposed to the same ad multiple times, with engagement frequencies ranging from 1 to 12 occurrences.***
> - ***After reviewing a sample of user engagements:***
>   - ***A near-duplicated pattern was noticed for users' multiple exposures***

- ***One user (ID: a100000882) exhibited 12 ad exposures:***
  - *Within these 12 records, several entries appeared repeated, with some columns identical while others showed slight variations.*
  - **For instance,* this user converted in 4 records — two under the personalized variant (served on January 14) and two under the control variant (served on January 18). The subscription dates associated with these conversions were either January 14 or January 18, suggesting minor inconsistencies or multiple conversions logged for the same user within a short period.*

- ***A similar pattern was observed for another sample user with 10 ad exposures (ID: a100000878):***
  - *The user interacted primarily through House Ads and Email channels.*
  - *Multiple records showed identical values across most columns, with slight variations in date_served and date_subscribed.*
  - *The user converted multiple times on the same channel (Email, control variant) — an unusual pattern since a user typically subscribes only once.*
  - *These repeated or inconsistent entries suggest potential data duplication or logging issues, where multiple impressions and conversions might have been recorded for a single actual event.*

- ***For a user with 8 recorded engagements (User ID: a100000875), several inconsistencies were observed:***
  - *Despite being recorded as the same user, their age group alternated between 19–24 years and 45–55 years, indicating a data entry or merge error.*
  - *The user converted four times. Subscription dates (1/7/18 and 1/11/18) were reused across records, often paired with inconsistent cancellation statuses.*
  - *Some records show the user as retained, while others mark them as canceled, even within the same channel and week.*
  - *This record highlights duplicated and conflicting user engagement logs, likely resulting from data integration or tracking issues.*

- ***Out of the 13 users who saw the ad 5 times, only 8 converted.***

Data Cleaning

$1-$ Removing Duplicates :

$a)$ Remove Exact Duplicates (37 duplicated raws)

$b)$ Remove Near-Duplicates to ensures each user's engagement on a given day with a specific ad type is counted only once.

In [479…

```
# 1- Remove exact duplicates
marketing.drop_duplicates(inplace=True)

# Checking:
print(duplicates(marketing))
```

```
The Dataset has no Duplicated Values with 10000 Row
```

In [480…

```
# 2- Remove near-duplicate records
subsets= ['user_id', 'date_served',      'marketing_channel', 'variant', 'converted'
marketing.drop_duplicates(subset=subsets,inplace=True, keep='first')
```

In [481…

```
marketing.describe()
```

Out[481…

	user_id	date_served	marketing_channel	variant	converted	language_displaye
count	9903	9887	9888	9903	9888	99(
unique	7309	31	5	2	2	
top	a100000882	1/15/18	House Ads	control	False	Engli:
freq	6	784	4655	5009	8853	96(

$2-$ Changing Dates Data Types :

$a)$ ***date_served:*** *str* to *date*

$b)$ ***date_subscribed:*** *str* to *date*

$c)$ ***date_canceled:*** *str* to *date*

In [482…

```
# Create a function to change the dates data types:
def date_change(df,col1,col2,col3):
    df[col1]= pd.to_datetime(df[col1])
    df[col2]= pd.to_datetime(df[col2])
    df[col3]= pd.to_datetime(df[col3])
    check= df.loc[:,[col1,col2,col3]].info()
    return check

date_change(marketing,'date_served','date_subscribed','date_canceled')
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 9903 entries, 0 to 10036
Data columns (total 3 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   date_served      9887 non-null   datetime64[ns]
 1   date_subscribed  1749 non-null   datetime64[ns]
 2   date_canceled    575 non-null    datetime64[ns]
dtypes: datetime64[ns](3)
memory usage: 309.5 KB
```

$3-$ Standardize Subscription Dates$:$

To ensure logical and consistent relationships between engagement and subscription dates, the following adjustments were applied:

$a)$ ***For converted users:***

- ***Issue:*** Some records show *date_subscribed* earlier than *date_served*. However, a user cannot subscribe before seeing the ad.
- ***Action:*** When *converted* = *True* and *date_subscribed* < *date_served*, replace *date_subscribed* with *date_served*.

$b)$ ***For not-converted users:***

- ***Issue:*** Some users who were exposed to multiple ads have a *date_subscribed* value recorded even when *converted* = *False*. This creates inconsistencies since non-converted records should not have a valid subscription date.
- ***Action:*** Set *date_subscribed* to *NaN* for all non-converted users to remove this confusion.

In [483…

```
# Standardize Subscription dates (Converted Users):
marketing['date_subscribed']=np.where(np.logical_and(marketing.converted == True,ma
                                      marketing['date_served'],
                                      marketing['date_subscribed'])

# Standardize Subscription dates (Non-Converted Users):
marketing['date_subscribed']=np.where(np.logical_and(marketing['converted'] == Fals
                                      pd.NaT,
                                      marketing['date_subscribed'])

# Changing the data type of date_subscribed
marketing['date_subscribed']=pd.to_datetime(marketing['date_subscribed'])

#Checking:
marketing.query("(converted == False & date_subscribed.notnull()) | (converted == T
```

Out[483…

user_id	date_served	marketing_channel	variant	converted	language_displayed	languag

4 − Handling Nulls :

a) Shared nulls across date_served (except for index 7038), marketing_channel, converted:

Since those columns share the same missing rows, dropping them together avoids keeping incomplete entries that would otherwise distort the analysis.

b) date_served (index 7038):

Since this is an isolated null in the middle of the dataset, forward-filling (ffill) after sorting by date is a reasonable strategy.

c) date_subscribed:

1. ***As a Precautionary measure,*** if the user converted,the missing values (if any) would be replaced with **date_served**.
2. If the user didn't convert, There is no need to handle the missing values as these values are naturally missing depending on whether the user subscribed or not.

d) date_canceled:

There is no need to handle its missing values in as these values are naturally missing depending on whether the user canceled his subscription or not. Filling them would introduce bias.

e) subscribing_channel

1. ***As a Precautionary measure,***
   - If the user converted,the missing values (if any) would be replaced with **marketing_channel**
   - If the user didn't convert & the subscribed channel isn't empty,then these values should be replaced with **NaN**.
2. If the user didn't convert, There is no need to handle the missing values as these values are naturally missing depending on whether the user subscribed or not .

f) is_retained:

1. ***As a Precautionary measure,***
   - If the user converted and there is no mention for canceling the subscription ,the missing values (if any) would be replaced with **True**.
   - If the user didn't convert & the is_retained value isn't empty,then these values should be replaced with **False**.
2. If the user didn't convert, There is no need to handle the missing values as these values are naturally missing depending on whether the user subscribed or not.

In [484…
```python
# Dropping Shared nulls across date_served (except for index 7038), marketing_chann
marketing.dropna(subset='marketing_channel', inplace=True)
```

In [485…
```python
# Nulls at date_served column
print(f'''
The date_served column has {marketing.date_served.isna().sum()} null value and its
```

```
The date_served column has 1 null value and its index is 7038
```

In [486…
```python
# date_served (index 7038):
# Sorting the table by date served
marketing=marketing.sort_values('date_served')

# Replacing null by forward fill method
marketing.date_served.fillna(method='ffill', inplace=True)

# Checking:
if marketing.date_served.isna().sum() == 0:
    print(f'\nThe Null values in date_served Column has been handeled, resulting in
else:
    print(f'''The date_served Column has {marketing.date_served.isna().sum()} NULL
```

```
The Null values in date_served Column has been handeled, resulting in a 0 Null Value
for this column
```

In [487…
```python
# date_subscribed:
marketing['date_subscribed']= np.where(np.logical_and(marketing.converted==True,mar
                                       marketing.date_served,marketing.date_subscri
```

In [488…
```python
# subscribing_channel:
marketing['subscribing_channel']= np.where(np.logical_and(marketing.converted==True
                                           marketing.marketing_channel,marketing.su

marketing['subscribing_channel']= np.where(np.logical_and(marketing.converted==Fals
                                           np.nan,marketing.subscribing_channel)
```

In [489…
```python
#is_retained:
marketing['is_retained']= np.where(np.logical_and(marketing.converted==True,marketi
                                   True,marketing.is_retained)

marketing['is_retained']= np.where(marketing.converted==False, np.nan,marketing.is_
```

In [490…
```python
# Checking for Null Values:
print(missing(marketing))
marketing.isna().sum().reset_index().rename(columns={'index':'Column_Name',0:'NULLs
```

```
The Dataset has 35872 NULL Values that are distributed as follows:
```

Out[490…

Column_Name	NULLs_Count
user_id	0
date_served	0
marketing_channel	0
variant	0
converted	0
language_displayed	0
language_preferred	0
age_group	0
date_subscribed	8853
date_canceled	9313
subscribing_channel	8853
is_retained	8853

$5-$ Handeling converted column :

– ***Changing Data Type:*** *str* to *boolean*

In [491…

```
# Changing the data type of converted:
marketing['converted']=marketing['converted'].astype('bool')
marketing['converted'].dtype
```

Out[491…

```
dtype('bool')
```

$6-$ Adjusting user id column :

Due to the inconsistences in some records, where a user may have more than one age group, it would be better to adjust the user name column by adding the first 2 charcters of age group values to the user id

In [492…

```
marketing['user_id']=[f'{x}-{y.split(" ")[0]}' for x,y in zip(marketing['user_id'],
#marketing['user_id']=[f'{x}-{y[0]}' for x,y in zip(marketing['user_id'],marketing[
marketing.head()
```

Out[492...

	user_id	date_served	marketing_channel	variant	converted	language_di
0	a100000029-0-18	2018-01-01	House Ads	personalization	True	
6678	a100004324-30-36	2018-01-01	House Ads	personalization	False	
6676	a100004323-24-30	2018-01-01	House Ads	personalization	False	
6674	a100004322-19-24	2018-01-01	House Ads	personalization	False	
6672	a100004321-0-18	2018-01-01	House Ads	personalization	False	

In [493...

```
marketing.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 9888 entries, 0 to 7038
Data columns (total 12 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   user_id             9888 non-null   object
 1   date_served         9888 non-null   datetime64[ns]
 2   marketing_channel   9888 non-null   object
 3   variant             9888 non-null   object
 4   converted           9888 non-null   bool
 5   language_displayed  9888 non-null   object
 6   language_preferred  9888 non-null   object
 7   age_group           9888 non-null   object
 8   date_subscribed     1035 non-null   datetime64[ns]
 9   date_canceled       575 non-null    datetime64[ns]
 10  subscribing_channel 1035 non-null   object
 11  is_retained         1035 non-null   object
dtypes: bool(1), datetime64[ns](3), object(8)
memory usage: 936.7+ KB
```

7 − Adding New Columns :

- ***is_house_ad:*** Identifies if a particular marketing asset was a house ad or not *(since it is the most frequent value in this column "4733 out of 10000")*
- ***matched_lang:*** conveys whether the ad was shown to the user in their preferred language
- ***dow:*** service Days starting from Monday till Sunday, t measure the most frequent days
- ***ad_repeated:*** to check whether the user saw the ad multiple times

In [494...

```
# Adding the is_house_ad Column
marketing['is_house_ad']=[
    True if x=="House Ads"
```

```
    else False for x in marketing.marketing_channel]

marketing.loc[:,['marketing_channel','is_house_ad']].sample(5)
```

Out[494…

	marketing_channel	is_house_ad
7261	House Ads	True
6899	Facebook	False
4201	House Ads	True
9028	Instagram	False
7772	Push	False

In [495…

```
# Adding matched_lang Column
marketing['matched_lang']=np.where(marketing['language_displayed']==marketing['lang

marketing.loc[:,['language_displayed','language_preferred','matched_lang']].sample(
```

Out[495…

	language_displayed	language_preferred	matched_lang
2169	English	English	True
1486	English	English	True
2357	English	English	True
632	English	English	True
3158	English	English	True

In [496…

```
# Adding dow column:
marketing['dow']= ['Mo' if x==0
                  else 'Tu' if x== 1
                  else 'We' if x==2
                  else 'Th' if x==3
                  else 'Fr' if x==4
                  else 'Sa' if x==5
                  else 'Su'for x in marketing.date_served.dt.dayofweek]

marketing.loc[:,['date_served','dow']].sample(5)
```

Out[496…

	date_served	dow
118	2018-01-06	Sa
3658	2018-01-28	Su
1778	2018-01-03	We
7192	2018-01-14	Su
8814	2018-01-08	Mo

In [497…

```
# Adding ad_repeated column:
is_repeated_user = marketing.user_id.value_counts()>1

repeated=is_repeated_user.reset_index().rename(columns={'index':'user_id','user_id'
                         .query('repeated==True').user_id.to_list()


marketing['ad_repeated']=[True if x in repeated else False for x in marketing.user_

marketing.loc[:,['user_id','ad_repeated']].head()
```

Out[497…

	user_id	ad_repeated
0	a100000029-0-18	False
6678	a100004324-30-36	True
6676	a100004323-24-30	True
6674	a100004322-19-24	True
6672	a100004321-0-18	True

6 – Mapping Values to Existing Columns:

> *Note*:
> *Due to the way pandas stores data, in a large dataset, it can be computationally inefficient to store columns of strings. In such cases, it can speed things up to instead store these values as numbers.*

- ***marketing_channel*** will be as follows:
  - *House Ads = 1*
  - *Push = 2*
  - *Facebook = 3*
  - *Instagram = 4*
  - *Email = 5*

In [498…

```
# Mapping marketing_channel column:
ch_dict={'House Ads' : 1,'Push' : 2,'Facebook' : 3,'Instagram' : 4,'Email' : 5}
marketing['ch_code']=marketing.marketing_channel.map(ch_dict).astype('Int64')
marketing.loc[:,['marketing_channel','ch_code']].info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 9888 entries, 0 to 7038
Data columns (total 2 columns):
 #   Column             Non-Null Count  Dtype 
---  ------             --------------  ----- 
 0   marketing_channel  9888 non-null   object
 1   ch_code            9888 non-null   Int64 
dtypes: Int64(1), object(1)
memory usage: 241.4+ KB
```

Data Exploring

Initial Investigation:

– ***Building Functions to automate analysis:***

- The goal is to reduce repeated codes in order to avoid Typos & Bugs

In [499…

```
# 1-Building a Function for counting users:
def counting (df,col_name):
    nums= marketing.groupby(col_name).user_id.count().reset_index()
    nums['Percentage']=nums.user_id/nums.user_id.sum()
    return nums
```

In [500…

```
# 2-Building a Function for unique users:
def uniques (df,col_name):
    distinct = marketing.groupby(col_name).user_id.nunique().reset_index()
    distinct['Percentage']=distinct.user_id/distinct.user_id.sum()
    return distinct
```

In [501…

```
# 3-Building a Function for Line Plots:
def line_plot(df,col1_name,col2_name):
    # Data
    x= df[col1_name].astype('str').to_list()
    y= df[col2_name]

    # Creating the Line Chart
    fig,ax =plt.subplots(figsize = (8,6))
    ax.plot(x,y,color='#805D87',marker = 'H', alpha=.8)

    # Customizing Chart
    plt.title('',fontsize=14,color='#454775')

    plt.xlabel(col1_name,fontsize=12,color='#313E4C')
    plt.xticks(rotation=90,fontsize=8,color='#415366')

    plt.ylabel(col2_name,fontsize=12,color='#313E4C')
    plt.yticks(fontsize=8,color='#415366')

    ax.spines['top'].set_visible(False)
```

```
    ax.spines['right'].set_visible(False)
    for spine in ax.spines.values():
        spine.set_linewidth(1.2)
        spine.set_edgecolor('#415366')
        spine.set_alpha(.8)

    # Data Annotation with values
    for i, v in enumerate(y):
      plt.text(i,v+15, f"{v:.0f}", ha='center', va='center',fontsize=7,color='#313E
```

In [502…

```
# 4-Building a Function for Bar Plots:
def bar_plot (df,col1_name,col2_name):
    # Data
    x= df[col1_name].to_list()
    y=df[col2_name]

    # Define bar colors based on performance
    colors = ['#805D87' if n == y.max() else '#94D1E7' for n in y]

    # Create the bar chart
    fig, ax= plt.subplots(figsize=(5,5))
    ax.bar(x,y, color=colors, alpha=.8)

    # Customizing Chart
    plt.title('',fontsize=12,color='#454775')

    plt.xlabel(col1_name,fontsize=10,color='#313E4C')
    plt.xticks(fontsize=8, color='#415366')

    plt.ylabel(col2_name,fontsize=10,color='#313E4C')
    plt.yticks(fontsize=8, color='#415366')

    ax.spines['top'].set_visible(False)
    ax.spines['right'].set_visible(False)
    for spine in ax.spines.values():
        spine.set_linewidth(1.2)
        spine.set_edgecolor('#415366')
        spine.set_alpha(.8)

    # Annotate bars with their values
    for i, v in enumerate(y):
      plt.text(i,v+.003, f"{v:.0f}", ha='center', va='bottom',fontsize=8,color='#31
```

In [503…

```
# 5-Building a Function for Pie Plots:
def pie_plot (df,col_name,col_label):
    # Defining colors based on performance
    colors = ['#805D87' if x == df[col_name].max() else '#94D1E7' for x in df[col_n

    #Data
    labels=df[col_label].str.capitalize().to_list()
    size = 0.45

    # Creating the Chart
    plt.subplots(figsize = (3.5,3.5))
    wedges, texts, autotexts=plt.pie(df[col_name], radius=1, colors= colors,labels
```

```
                                    textprops={'fontsize': 10,'color':'#313E4C'}, wedg

    # Customizing Chart
    for w in wedges:
        w.set_alpha(0.8)

    plt.title('',fontsize=12,color='#454775')
```

In [504…

```
# 6-Building a Function for Horizontal stacked Plots:
def stackedh_plot(df,col1_name,col2_name,col3_name):
    # Data
    x=df[col1_name].to_list()
    y=df[col2_name]
    z=df[col3_name]

    # Creating the Chart
    fig,ax= plt.subplots(figsize=(5,5))
    bin_size=.5
    ax.barh(x,y, label=col2_name,color='#805D87',alpha=.8)
    ax.barh(x,z,bin_size,label=col3_name, color='#94D1E7',alpha=.8)

    # Chart Customization
    plt.title('', fontsize=12,color='#454775')

    plt.xlabel('', fontsize=10,color='#313E4C')
    plt.xticks(fontsize=8,color='#415366')

    plt.ylabel(col1_name, fontsize=10,color='#313E4C')
    plt.yticks(fontsize=8,color='#415366')

    plt.legend(fontsize=9,labelcolor='#313E4C',loc='center right',fancybox=True, sh

    ax.spines['top'].set_visible(False)
    ax.spines['right'].set_visible(False)
    for spine in ax.spines.values():
        spine.set_linewidth(1.2)
        spine.set_edgecolor('#415366')
        spine.set_alpha(.8)
```

In [505…

```
# 7-Building a Function for Horizontal Bar Plots:
def hbar_plot(df, col1_name,col2_name):
    # Data
    x= df[col1_name].to_list()
    y=df[col2_name]

    # Defining colors based on performance
    colors = ['#805D87' if n == y.max() else '#94D1E7' for n in y]

    # Creating the chart
    fig, ax= plt.subplots(figsize=(5,5))
    ax.barh(x,y,alpha=.8,color=colors)

    # Customizing the Chart
    plt.title('', fontsize=12,color='#454775')
```

```
    plt.xlabel(col2_name, fontsize=10,color='#313E4C')
    plt.xticks(fontsize=8, color='#415366')

    plt.ylabel(col1_name, fontsize=10,color='#313E4C')
    plt.yticks(fontsize=8, color='#415366')

    ax.spines['top'].set_visible(False)
    ax.spines['right'].set_visible(False)
    for spine in ax.spines.values():
        spine.set_linewidth(1.2)
        spine.set_edgecolor('#415366')
        spine.set_alpha(.8)

    # Annotating bars with values
    for i,v in enumerate(y):
        plt.text(v,i,v,va='center',ha='left',fontsize=8,color='#313E4C')
```

1 – *Number of Daily users:*

In [506…

```
# Number of daily users :
daily_users = counting(marketing,['date_served','dow']).rename(columns={'date_serve

daily_users['Date']=daily_users.Date.dt.date

daily_users.style.hide().format({'Percentage':'{:,.2%}'})
```

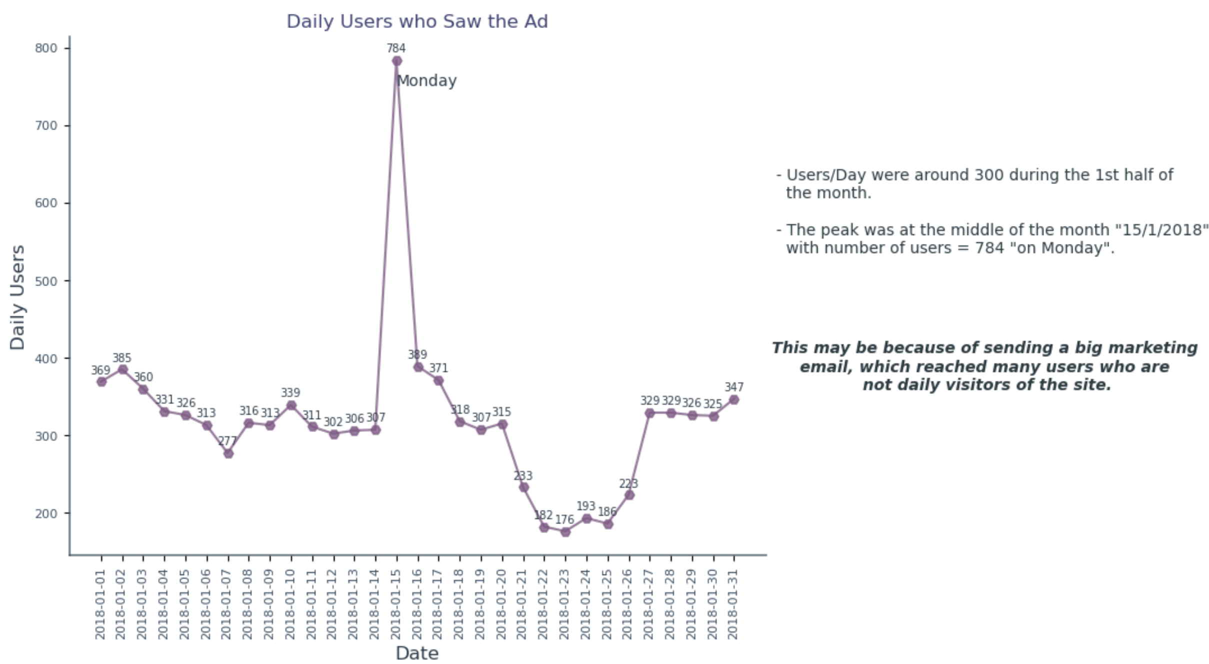
Out[506…

Date	dow	Daily Users	Percentage
2018-01-01	Mo	369	3.73%
2018-01-02	Tu	385	3.89%
2018-01-03	We	360	3.64%
2018-01-04	Th	331	3.35%
2018-01-05	Fr	326	3.30%
2018-01-06	Sa	313	3.17%
2018-01-07	Su	277	2.80%
2018-01-08	Mo	316	3.20%
2018-01-09	Tu	313	3.17%
2018-01-10	We	339	3.43%
2018-01-11	Th	311	3.15%
2018-01-12	Fr	302	3.05%
2018-01-13	Sa	306	3.09%
2018-01-14	Su	307	3.10%
2018-01-15	Mo	784	7.93%
2018-01-16	Tu	389	3.93%
2018-01-17	We	371	3.75%
2018-01-18	Th	318	3.22%
2018-01-19	Fr	307	3.10%
2018-01-20	Sa	315	3.19%
2018-01-21	Su	233	2.36%
2018-01-22	Mo	182	1.84%
2018-01-23	Tu	176	1.78%
2018-01-24	We	193	1.95%
2018-01-25	Th	186	1.88%
2018-01-26	Fr	223	2.26%
2018-01-27	Sa	329	3.33%
2018-01-28	Su	329	3.33%
2018-01-29	Mo	326	3.30%
2018-01-30	Tu	325	3.29%

	Date	dow	Daily Users	Percentage
	2018-01-31	We	347	3.51%

In [507...

```
# Visualization - Number of daily users :
line_plot(daily_users,'Date','Daily Users')

# Additional Customization
plt.title('Daily Users who Saw the Ad')

# Findings
text_d_u =f'''
- Users/Day were around 300 during the 1st half of
  the month.\n
- The peak was at the middle of the month "15/1/2018"
  with number of users = {daily_users['Daily Users'].max()} "on Monday".'''

text2_d_u='''
This may be because of sending a big marketing
email, which reached many users who are
not daily visitors of the site.'''

plt.text(32,600,text_d_u,va='center',ha='left',color='#313E45')
plt.text(42,400,text2_d_u,va='center',ha='center',color='#313E45',fontstyle='italic
plt.text('2018-01-15', 767, 'Monday',va='top', ha='left',color='#313E4C');
```



2 − *Number of Weekday users:*

In [508...

```
# Number of weekday users :
weekday_users = counting(marketing,['dow']).rename(columns={'dow':'Day','user_id':'
                        .sort_values('# Users', ascending=False)
```

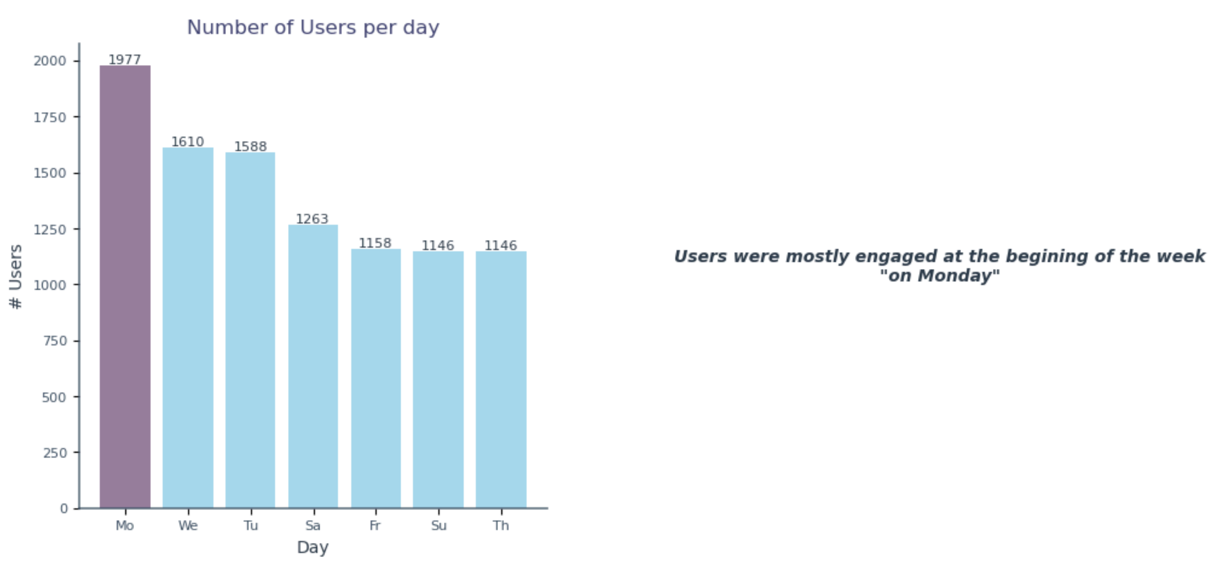
```
weekday_users.style.hide().format({'Percentage':'{:,.2%}'})
```

Out[508…

Day	# Users	Percentage
Mo	1977	19.99%
We	1610	16.28%
Tu	1588	16.06%
Sa	1263	12.77%
Fr	1158	11.71%
Su	1146	11.59%
Th	1146	11.59%

In [509…

```
# Visualization - Number of weekday users:
bar_plot(weekday_users,'Day','# Users')

# Additional Customization
plt.title('Number of Users per day')

# Findings
text_w = '''
Users were mostly engaged at the begining of the week\n"on Monday"'''
plt.text(13,1000,text_w,va='bottom',ha='center' ,color='#313E4C',fontstyle='italic'
```



3 − *Number of users according to variant classification:*

In [510…

```
# Number of users according to variant categories
var_users = counting(marketing,'variant').rename(columns={'user_id':"num_users"})

var_users.style.hide().format({'Percentage':'{:,.2%}'})
```

Out[510…

variant	num_users	Percentage
control	4994	50.51%
personalization	4894	49.49%

In [511…

```
# Visualization - Number of users according to variant categories:
pie_plot(var_users,'num_users','variant')

# Additional Customization
plt.title('Number of users according to variant categories')

# Findings
text_v = '''
Users were almost evenly assigned between the control group \nand the personalizati
plt.text(5,0,text_v,ha='center',va='bottom',fontsize = 10, weight = 'semibold',font
```

4 — *Number of converted users vs. non-converted users:*

In [512…

```
# Number of converted users vs. non-converted users
converted_users = counting(marketing,'converted').rename(columns={'converted':'stat

converted_users['status'] = np.where(converted_users['status']==True,'Converted','N

converted_users.style.hide().format({'Percentage':'{:,.2%}'})
```

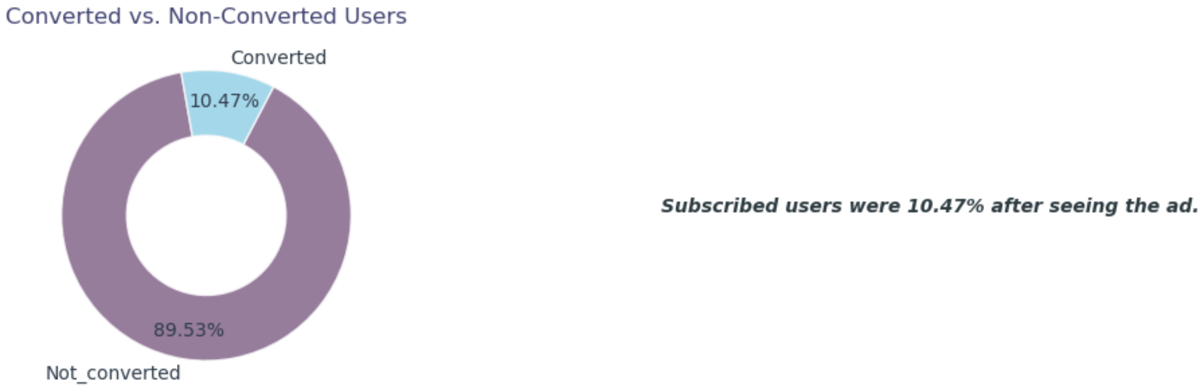
Out[512…

status	num_users	Percentage
Not_Converted	8853	89.53%
Converted	1035	10.47%

In [513…

```
# Visualization - Number of converted users vs. non-converted users:
pie_plot(converted_users,'num_users','status')

# Additional Customization
plt.title('Converted vs. Non-Converted Users')

# Findings
text_c = f'''
```

```
Subscribed users were {converted_users.Percentage.min():.2%} after seeing the ad.''
plt.text(5,0,text_c,ha='center',va='bottom',fontsize = 10, weight = 'semibold',font
```



5 – *Displayed Lanaguage vs. Preferred Language:*

In [514…

```
# Displayed Lanaguage vs. Preferred Language
lang_displayed=counting(marketing,'language_displayed').rename(columns={'language_d

lang_preferred=counting(marketing,'language_preferred').rename(columns={'language_p

lang=lang_displayed.merge(lang_preferred,on='Language',suffixes=('_Displayed','_Pre

lang.style.hide().format({"Percentage_Displayed":"{:,.2%}","Percentage_Preferred":"
```

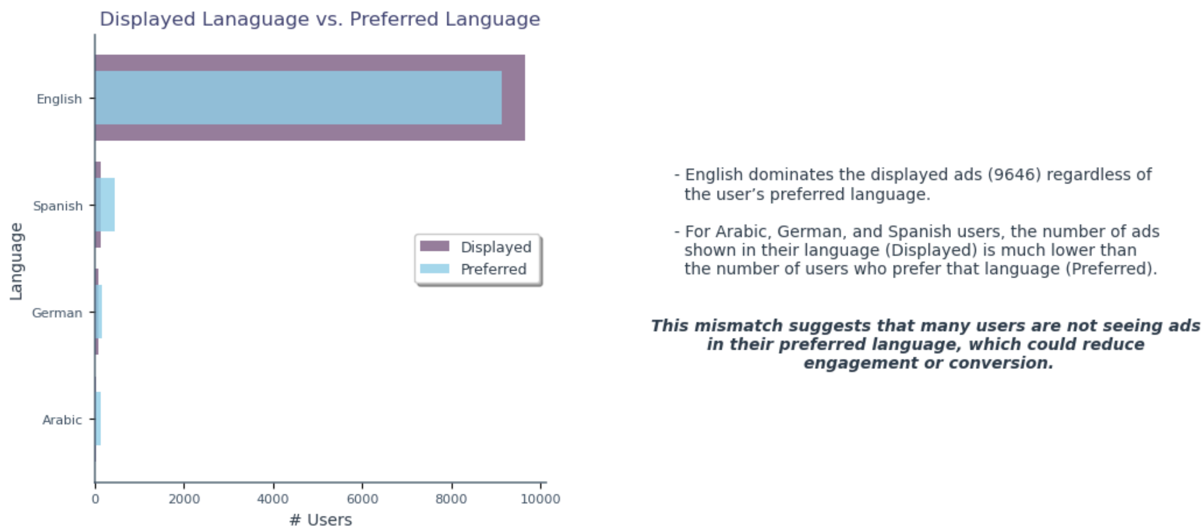
Out[514…

Language	Displayed	Percentage_Displayed	Preferred	Percentage_Preferred
Arabic	27	0.27%	145	1.47%
German	80	0.81%	165	1.67%
Spanish	135	1.37%	446	4.51%
English	9646	97.55%	9132	92.35%

In [515…

```
# Visualization - Displayed Lanaguage vs. Preferred Language:
stackedh_plot(lang,'Language','Displayed','Preferred')

# Additional Customization
plt.title('Displayed Lanaguage vs. Preferred Language')
plt.xlabel('# Users')

# Findings
text_l=f'''
- English dominates the displayed ads ({lang.Displayed.max()}) regardless of \n  th
- For Arabic, German, and Spanish users, the number of ads \n  shown in their langu
  the number of users who prefer that language (Preferred).\n
'''
text2_l='''
This mismatch suggests that many users are not seeing ads
in their preferred language, which could reduce
```

```
engagement or conversion.
'''
plt.text(13000,1,text_l,color='#313E4C'),
plt.text(18700,.3,text2_l, ha='center',fontstyle='italic',weight='semibold', fontsi
```



6 – *Distribution of age among users:*

In [516…

```
# Distribution of age among users
age_distribution = uniques(marketing,'age_group').rename(columns={'age_group':'Age
age_distribution['Age Group']= age_distribution['Age Group'].replace(r' years', '',
age_distribution.style.hide().format({'Percentage':'{:,.2%}'})
```

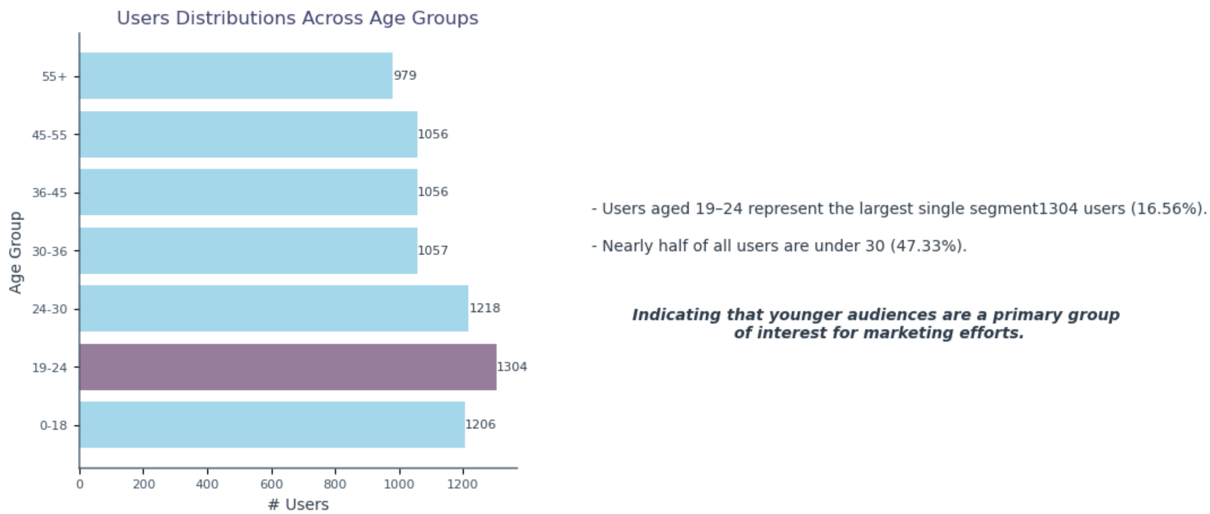
Out[516…

Age Group	# Users	Percentage
0-18	1206	15.31%
19-24	1304	16.56%
24-30	1218	15.46%
30-36	1057	13.42%
36-45	1056	13.41%
45-55	1056	13.41%
55+	979	12.43%

In [517…

```
# Visualization - Distribution of age among users:
hbar_plot(age_distribution,'Age Group','# Users')

# Additional Customization
plt.title('Users Distributions Across Age Groups')

# Findings
text_age_d= f'''
- Users aged 19–24 represent the largest single segment{age_distribution['# Users']
```

```
- Nearly half of all users are under 30 ({age_distribution.iloc[:3,2].sum():.2%}).'

text2_age_d='''
Indicating that younger audiences are a primary group \nof interest for marketing e

plt.text(1600,3,text_age_d,color='#313E4C')
plt.text(2500,1.5,text2_age_d, ha='center',fontstyle='italic', weight='semibold', f
```



7 – *Marketing Channels:*

In [518...

```
# Number of users for each marketing channel:
ch_users=counting(marketing,'marketing_channel')\
                   .rename(columns={'marketing_channel':'Marketing Channel','user_i
                   .sort_values('# Users')

ch_users.style.hide().format({'Percentage':'{:.2%}'})
```

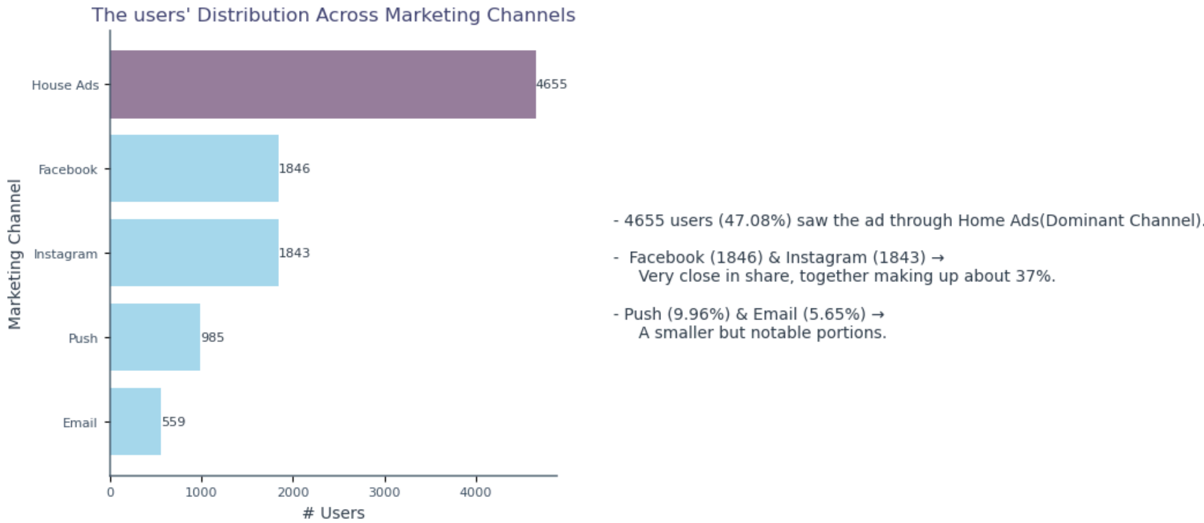
Out[518...

Marketing Channel	# Users	Percentage
Email	559	5.65%
Push	985	9.96%
Instagram	1843	18.64%
Facebook	1846	18.67%
House Ads	4655	47.08%

In [519...

```
# Visualization - Number of users for each marketing channel:
hbar_plot(ch_users,'Marketing Channel','# Users')

# Additional Customization
plt.title("The users' Distribution Across Marketing Channels")

# Findings
text_ch_u=f'''
```

```
- {ch_users['# Users'].max()} users ({ch_users.Percentage.max():.2%}) saw the ad th
-  {ch_users.iloc[3,0]} ({ch_users.iloc[3,1]}) & {ch_users.iloc[2,0]} ({ch_users.il
     Very close in share, together making up about {ch_users.iloc[3,2]+ch_users.ilo
- Push ({ch_users.iloc[1,2]:.2%}) & Email ({ch_users.iloc[0,2]:.2%}) →
     A smaller but notable portions.'''
plt.text(5500,1,text_ch_u,color='#313E4C');
```



In [520…

```
# Subscribing Channels
sub_channel=uniques(marketing,'subscribing_channel').\
                  rename(columns={'subscribing_channel':'Subscribing Channel','user
                  sort_values('# Subscribers')

sub_channel.style.hide().format({'Percentage':'{:,.2%}'})
```

Out[520…

Subscribing Channel	# Subscribers	Percentage
Push	78	7.61%
Email	187	18.24%
Facebook	224	21.85%
Instagram	238	23.22%
House Ads	298	29.07%

In [521…

```
# Retained Subscribers
retained = marketing.query('is_retained == True').groupby('subscribing_channel').us
                     .rename(columns={'subscribing_channel':'Subscribing Channel','u
                      sort_values('# Retained')

retained.style.hide()
```

Out[521…

Subscribing Channel	# Retained
Push	53
Email	143
Facebook	153
Instagram	154
House Ads	173

In [522…

```
# Merging subscribing channeles and retained subscribers
subscribers = sub_channel.merge(retained,on='Subscribing Channel').iloc[:,[0,1,3]]

subscribers['Retained Percentage']=subscribers['# Retained']/subscribers['# Subscri

subscribers['Middle Point']=(subscribers['# Subscribers']/2)

subscribers.style.hide().format({'Retained Percentage':'{:.2%}','Middle Point':'{:.
```

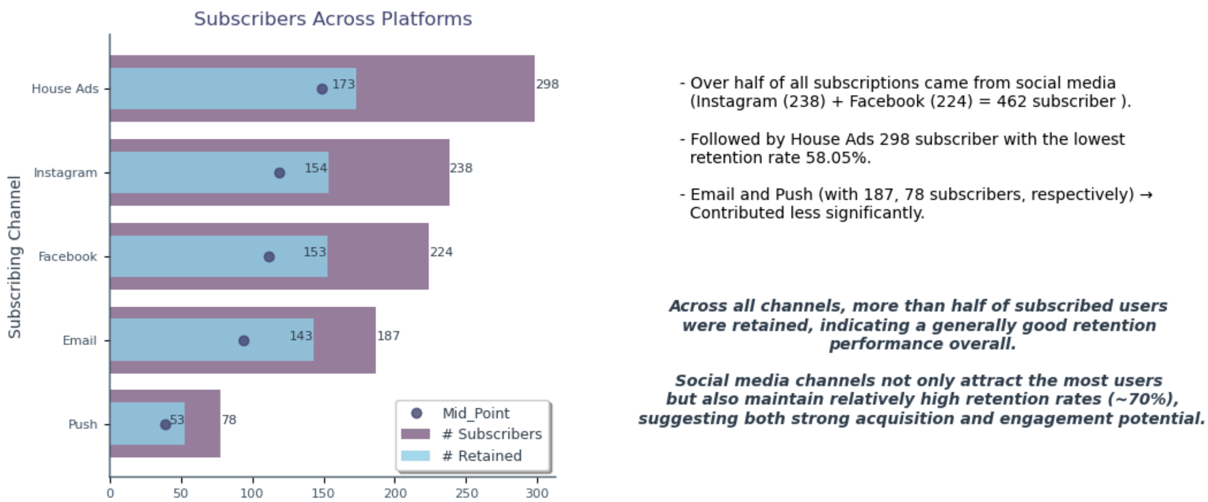
Out[522…

Subscribing Channel	# Subscribers	# Retained	Retained Percentage	Middle Point
Push	78	53	67.95%	39
Email	187	143	76.47%	94
Facebook	224	153	68.30%	112
Instagram	238	154	64.71%	119
House Ads	298	173	58.05%	149

In [523…

```
# Visualization - subscribing channeles and retained subscribers
stackedh_plot(subscribers,'Subscribing Channel','# Subscribers','# Retained')

mid_point=subscribers['Middle Point']
plt.scatter(mid_point,subscribers['Subscribing Channel'].to_list(), label='Mid_Poin

# Additional Customization
plt.title('Subscribers Across Platforms')
plt.legend(fontsize=9,labelcolor='#313E4C',loc='lower right',fancybox=True, shadow=

# Annotating bars with values
for i,v in enumerate(subscribers['# Subscribers']):
    plt.text(v,i,v,fontsize=8, color='#313E4C')

for i,v in enumerate(subscribers['# Retained']):
    plt.text(v,i,v,ha='right',fontsize=8, color='#313E4C')

# Findings
text_sub=f'''
- Over half of all subscriptions came from social media
  (Instagram ({subscribers.iloc[3,1]}) + Facebook ({subscribers.iloc[2,1]}) = {subs
- Followed by House Ads {subscribers.iloc[4,1]} subscriber with the lowest \n  rete
- Email and Push (with {subscribers.iloc[1,1]}, {subscribers.iloc[0,1]} subscribers
```

```
'''

text2_sub='''
Across all channels, more than half of subscribed users \nwere retained, indicating
Social media channels not only attract the most users \nbut also maintain relativel

plt.text(400,2,text_sub)
plt.text(570,0,text2_sub, ha='center',fontstyle='italic',weight='semibold', fontsiz

plt.show()
```



Influence Factors :

$Q1$: *What factors most stringly influence user Conversion and Retention Rates?*

– *Building Functions to automate analysis:*

In [524...

```
# 1- Conversion & Retention Rates function:
def con_ret (df,df2,cols,target):
    first= df.groupby(cols)[target].nunique().reset_index()
    second=df2.groupby(cols)[target].nunique().reset_index()
    result=first.merge(second,on= cols, suffixes=('_total','_part'))
    result['Rate']= round(result.iloc[:,-1]/result.iloc[:,-2],4)
    result=result.sort_values('Rate', ascending = False)
    result.columns = [x.replace('_', ' ').title() if x in cols else x for x in resu
    return result
```

In [525...

```
# 2- Comparison between Conversion & Retention Rates function:
def comparison (df,df2,df3,cols,target):
    first= df.groupby(cols)[target].nunique().reset_index()
    second=df2.groupby(cols)[target].nunique().reset_index()
    result1=first.merge(second,on= cols)
    result1['Conversion Rate']= round(result1.iloc[:,-1]/result1.iloc[:,-2],4)
```

```python
    third = df3.groupby(cols)[target].nunique().reset_index()
    result2= second.merge(third,on= cols)
    result2['Retention Rate']= round(result2.iloc[:,-1]/result2.iloc[:,-2],4)

    required_cols=list(cols)+ ['Conversion Rate','Retention Rate']

    final_result= result1.merge(result2, on=cols).loc[:,required_cols]
    final_result.columns = [x.replace('_', ' ').title() if x in cols else x for x i
    final_result= final_result.sort_values('Conversion Rate', ascending = False)

    return final_result
```

In [526…

```python
# 3- Bar Plot function:
def bars (df,col1,col2,rate):
    # Data
    x= df[col1].astype('str').apply(lambda x: x.title()).to_list()
    y= df[col2]

    # Defining colors based on performance
    colors = ['#805D87' if n > rate else '#94D1E7' for n in y]

    # Creating the chart
    fig, ax =plt.subplots(figsize=(4.5,4.5))
    ax.bar(x,y,width=.5, color=colors, alpha=.8)
    ax.axhline(y=rate, color='#454775', linestyle='--', linewidth=1, label='Overall

    # Customizing the chart
    plt.title('', fontsize=12,color='#454775')

    plt.xlabel('\n'+col1+'\n', fontsize=10, color='#313E4C')
    ax.tick_params(axis='x', color='#415366', labelcolor='#415366',labelsize=8)
    plt.ylabel('\n'+col2+'\n', fontsize=10, color='#313E4C')
    ax.tick_params(axis='y', color='#415366', labelcolor='#415366',labelsize=8)

    plt.legend(fontsize=9,labelcolor='#313E4C',loc='best',alignment='center', fancy

    ax.spines['top'].set_visible(False)
    ax.spines['right'].set_visible(False)
    for spine in ax.spines.values():
        spine.set_linewidth(1.2)
        spine.set_edgecolor('#415366')
        spine.set_alpha(.8)

    # Annotating chart with values
    plt.text(x[-1],rate, f'\u2003\u2003\u2003{rate:.2%}', ha= 'left', va ='bottom',

    for i,v in enumerate(y):
        plt.text(i,v+.005,f'{v:.2%}',va='bottom',ha='center', fontsize=8,color='#31
```

In [527…

```python
# 4- Combo Chart function:
def combo (df,col1,col2,col3,rate1,rate2):
    # Data
    x = df[col1].apply(lambda x: x.title()).to_list()
    y = df[col2]
    z = df[col3]
```

```python
    # Defining colors based on performance
    colors1 = ['#805D87' if n > rate1 else '#94D1E7' for n in y]
    colors2 = ['#454775' if m > rate2 else '#EA9FBB' for m in z]

    # Creating the chart
    fig,ax1=plt.subplots(figsize=(5,5))

    # 1- Bar Plot
    ax1.bar(x,y,width=.5, alpha=.8, color=colors1)

    # 2- Line & Scatter Plots
    ax2 = ax1.twinx()
    ax2.plot(x,z, ls='dotted', color='#51687F', label='Retention Rate', alpha=.5)
    ax2.scatter(x,z,color=colors2)

    # Customizing the chart
    plt.title('', fontsize=12, color='#454775')

    ax1.set_xlabel('\n'+col1+'\n', fontsize=10,color='#313E4C')
    ax1.tick_params(axis='x',labelcolor='#415366',labelsize=8)

    ax1.set_ylabel('\n'+col2+'\n', fontsize=10,color='#313E4C')
    ax1.tick_params(axis='y',labelcolor='#415366',labelsize=8)
    ax1.yaxis.set_major_formatter(mticker.PercentFormatter(1,decimals=False))

    ax2.set_ylabel('\n'+col3+'\n', fontsize=10, color='#313E4C')
    ax2.tick_params(axis='y',labelcolor='#415366',labelsize=8)
    ax2.yaxis.set_major_formatter(mticker.PercentFormatter(1,decimals=False))

    ax1.spines['top'].set_visible(False)
    ax2.spines['top'].set_visible(False)
    for spine in ax1.spines.values():
        spine.set_linewidth(1.2)
        spine.set_edgecolor('#415366')
        spine.set_alpha(.8)
    for spine in ax2.spines.values():
        spine.set_linewidth(1.2)
        spine.set_edgecolor('#415366')
        spine.set_alpha(.8)

    # Legend
    above_cr = mpatches.Patch(color='#805D87', label=f'Conversion Rate > {rate1:.2%
    below_cr = mpatches.Patch(color='#94D1E7', label=f'Conversion Rate ≤ {rate1:.2%
    above_rr = mlines.Line2D([], [], color='#454775', marker='o',linestyle='None',l
    below_rr = mlines.Line2D([], [], color='#EA9FBB', marker='o', linestyle='None',
    plt.legend(handles=[above_cr,below_cr,above_rr,below_rr],fontsize=8,labelcolor=
    bbox_to_anchor=(1.6, 1),alignment='center', fancybox=True, shadow=True,)
```

In [528…

```python
# 4- Horizontal Bar Chart function:
def h_bar(df,col1,col2,rate):
    # Data
    x= df[col1].astype('str').to_list()
    y= df[col2]

    # Defining colors based on performance
```

```
    colors = ['#805D87' if n > rate else '#94D1E7' for n in y]

    # Creating the chart
    fig, ax = plt.subplots(figsize = (6.5,6.5))
    ax.barh(x,y,.85, color=colors,alpha = .8)
    ax.axvline(x=rate, color='#454775', linestyle='--', linewidth=1, label='Overall

    # Customizing the chart
    plt.gca().invert_yaxis()

    plt.title('', fontsize=12, color='#454775')

    plt.xlabel(col2, fontsize=10, color='#313E4C')
    plt.xticks(fontsize=8, color='#415366')

    plt.ylabel(col1, fontsize=10, color='#313E4C')
    plt.yticks( fontsize=8, color='#415366')

    plt.legend(fontsize=8,labelcolor='#313E4C', loc='upper right', fancybox=True, s

    ax.spines['top'].set_visible(False)
    ax.spines['right'].set_visible(False)
    for spine in ax.spines.values():
        spine.set_linewidth(1.2)
        spine.set_edgecolor('#415366')
        spine.set_alpha(.8)

    # Annotating chart with values
    plt.text(rate,-1.01, f'{rate:.2%}', ha= 'left', va ='center', fontsize=9, color

    for i,v in enumerate(y):
        plt.text(v,i,f'{v:.2%}',va='center',ha='right', fontsize=6.5, color='#313E4
```

Conversion & Retention Rates

In [529…

```
# 1- Overall Conversion Rate
# Creating converted_users table:
converted_users = marketing.query('converted == True')

# Calculating The Overall Conversion Rate
converted=converted_users.user_id.nunique()
total_users = marketing.user_id.nunique()

conversion_rate = converted/total_users

print(f'\nThe Overall Conversion Rate = {round(conversion_rate*100,2)}%\n')
```

```
The Overall Conversion Rate = 13.01%
```

In [530…

```
# 2- Overall Retention Rate (spaning 1 month)
# Creating retained_users table:
retained_users = converted_users.query('is_retained == True')
```

```python
# Calculating The Overall Retention Rate
retained=retained_users.user_id.nunique()

retention_rate = retained/converted

print(f'\nThe Overall Retention Rate  = {round(retention_rate*100,2)}%\n')
```

```
The Overall Retention Rate  = 65.95%
```

1 − *Marketing Channels:*

In [531…

```python
# Calculating Conversion Rate across marketing channels
conversion_ch =con_ret(marketing,converted_users,'marketing_channel','user_id')
conversion_ch.columns=['Marketing Channel','Total Users','Converted','Conversion Ra
conversion_ch.style.hide().format({'Conversion Rate':'{:,.2%}'})
```

Out[531…

Marketing Channel	Total Users	Converted	Conversion Rate
Email	554	187	33.75%
Instagram	1798	238	13.24%
Facebook	1795	224	12.48%
Push	982	78	7.94%
House Ads	4025	298	7.40%

In [532…

```python
# Calculating Retention Rate across marketing channels
retention_ch=con_ret(converted_users,retained_users,'marketing_channel','user_id')
retention_ch.columns=['Marketing Channel','Converted','Retained','Retention Rate']
retention_ch.style.hide().format({'Retention Rate':'{:,.2%}'})
```

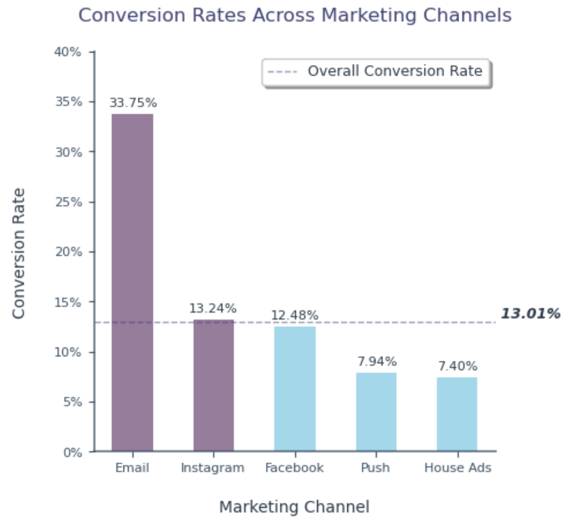
Out[532…

Marketing Channel	Converted	Retained	Retention Rate
Email	187	143	76.47%
Facebook	224	153	68.30%
Push	78	53	67.95%
Instagram	238	154	64.71%
House Ads	298	173	58.05%

In [533…

```python
# 1- Visualization - Conversion Rates Across Marketing Channels:
bars(conversion_ch,'Marketing Channel','Conversion Rate',conversion_rate)

# Additional Customization
plt.title('\nConversion Rates Across Marketing Channels\n')
plt.yticks(np.arange(0,.45,.05),[f'{y:.0%}' for y in np.arange(0,.45,.05)])

# Findings
```

```
text_conv_ch=f'''
- Email had the highest conversion rate ({conversion_ch['Conversion Rate'].max():.2
  the least used marketing channel (only {ch_users.Percentage.min():.2%} of total a
- House Ads, while accounting for the largest share of
  ad impressions ({ch_users.Percentage.max():.2%}), showed the lowest conversion
  rate ({conversion_ch['Conversion Rate'].min():.2%}), indicating poor effectivenes

text2_conv_ch='''
This contrast highlights that while Email campaigns are highly
effective,the heavy reliance on House Ads may not be an
efficient use of resources. \n
A redistribution of ad exposure toward higher-performing
channels could improve overall conversion results.'''

plt.text(6,.2,text_conv_ch, color='#313E4C')
plt.text(9.5,.06,text2_conv_ch, ha='center',fontstyle='italic',weight='semibold', f

plt.show();


# 2- Visualization - Retention Rates Across Marketing Channels:
bars(retention_ch,'Marketing Channel','Retention Rate',retention_rate)

# Additional Customization
plt.title('\nRetention Rates Across Marketing Channels\n')
plt.yticks(np.arange(0,1,.1),[f'{y:.0%}' for y in np.arange(0,1,.1)])

# Findings
text_ret_ch=f'''
- Email had the highest retention rate ({retention_ch['Retention Rate'].max():.2%})
  the least used marketing channel (only {ch_users.Percentage.min():.2%} of total a
- House Ads, while accounting for the largest share of ad
  impressions ({ch_users.Percentage.max():.2%}), showed the lowest retention rate
  ({retention_ch['Retention Rate'].min():.2%}), indicating weak post-subscription e
  relative to its reach.\n
'''

text2_ret_ch=f'''
All other marketing channels maintained retention rates above
or close to the overall average ({retention_rate:.2%}), highlighting
House Ads as the main underperforming channel'''

plt.text(6,.4,text_ret_ch, color='#313E4C')
plt.text(9.5,.2,text2_ret_ch, ha='center',fontstyle='italic',weight='semibold', fon

plt.show();
```

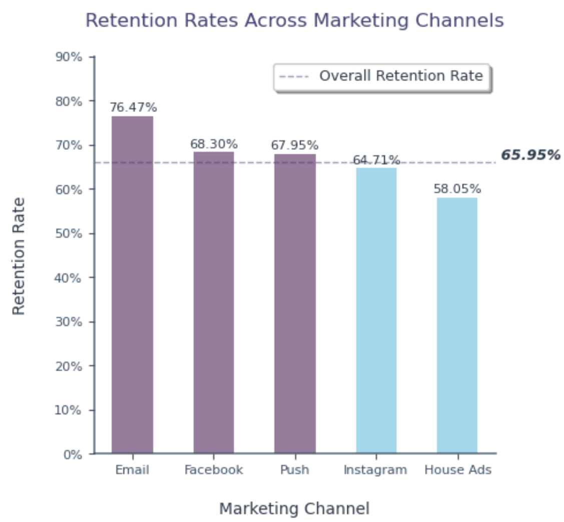


- Email had the highest conversion rate (33.75%), despite being the least used marketing channel (only 5.65% of total ads).

- House Ads, while accounting for the largest share of ad impressions (47.08%), showed the lowest conversion rate (7.40%), indicating poor effectiveness relative to its reach.

This contrast highlights that while Email campaigns are highly effective,the heavy reliance on House Ads may not be an efficient use of resources.

A redistribution of ad exposure toward higher-performing channels could improve overall conversion results.



- Email had the highest retention rate (76.47%), despite being the least used marketing channel (only 5.65% of total ads).

- House Ads, while accounting for the largest share of ad impressions (47.08%), showed the lowest retention rate (58.05%), indicating weak post-subscription engagement relative to its reach.

All other marketing channels maintained retention rates above or close to the overall average (65.95%), highlighting House Ads as the main underperforming channel

In [534…

```
# Comparing Conversion Rates with Retention Rates Across Marketing Channels:
performance_ch= comparison(marketing,converted_users,retained_users,['marketing_cha

performance_ch.style.hide().format({'Conversion Rate':'{:,.2%}','Retention Rate':'{
```

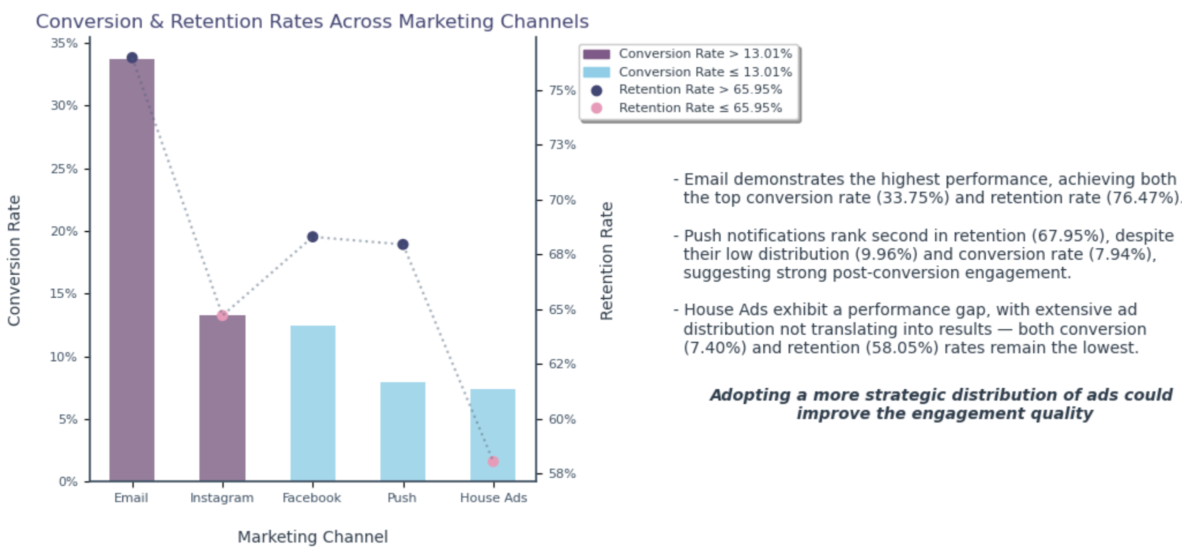
Out[534…

Marketing Channel	Conversion Rate	Retention Rate
Email	33.75%	76.47%
Instagram	13.24%	64.71%
Facebook	12.48%	68.30%
Push	7.94%	67.95%
House Ads	7.40%	58.05%

In [535…

```
# Visualization - Comparing Conversion Rates with Retention Rates Across Marketing
combo(performance_ch,'Marketing Channel','Conversion Rate','Retention Rate', conver

# Additional Customization
plt.title('Conversion & Retention Rates Across Marketing Channels')
```

```
# Findings
text_perform_ch=f'''
- Email demonstrates the highest performance, achieving both
  the top conversion rate ({performance_ch['Conversion Rate'].max():.2%}) and reten
- Push notifications rank second in retention ({performance_ch.iloc[3,2]:.2%}), des
  their low distribution ({ch_users[ch_users['Marketing Channel']=="Push"].iloc[0,2
  suggesting strong post-conversion engagement.\n
- House Ads exhibit a performance gap, with extensive ad
  distribution not translating into results — both conversion
  ({performance_ch['Conversion Rate'].min():.2%}) and retention ({performance_ch['R

text2_perform_ch='''
Adopting a more strategic distribution of ads could
improve the engagement quality'''

plt.text(6,.63,text_perform_ch, color='#313E4C')
plt.text(9,.6,text2_perform_ch, ha='center',fontstyle='italic',weight='semibold', f

plt.show();
```



2 − *Variant Classification:*

In [536…

```
# Calculating Conversion Rate across variant classifications
conversion_var =con_ret(marketing,converted_users,'variant','user_id')

conversion_var.columns=['Ad Classification','Total Users','Converted','Conversion R

conversion_var.style.hide().format({'Conversion Rate':'{:,.2%}'})
```

Out[536…

Ad Classification	Total Users	Converted	Conversion Rate
personalization	4089	687	16.80%
control	3844	346	9.00%

In [537…

```
# Calculating Retention Rate witnin Variant Classifications
retention_var =con_ret(converted_users,retained_users,'variant','user_id')

retention_var.columns=['Ad Classification','Converted','Retained','Retention Rate']

retention_var.style.hide().format({'Retention Rate':'{:,.2%}'})
```

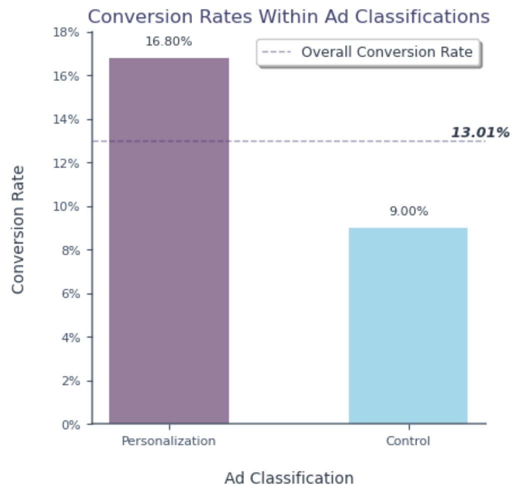
Out[537…

Ad Classification	Converted	Retained	Retention Rate
control	346	234	67.63%
personalization	687	450	65.50%

In [538…

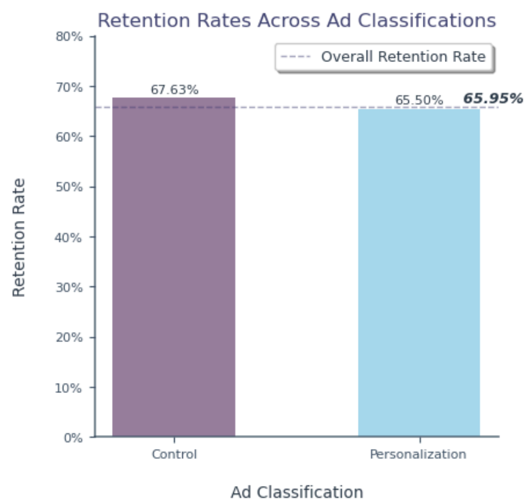
```
# 1- Visualization - Conversion Rates Within Variant Classifications:
bars(conversion_var,'Ad Classification','Conversion Rate',conversion_rate)

# Additional Customization
plt.title('Conversion Rates Within Ad Classifications')
plt.yticks(np.arange(0,.2,.02),[f'{y:.0%}' for y in np.arange(0,.2,.02)])

# Findings
text_con_var=f'''
- Personalized ads achieved a higher conversion rate ({conversion_var['Conversion R
  than both the control group ({conversion_var['Conversion Rate'].min():.2%}) and t
  rate ({conversion_rate:.2%}).\n'''

text2_con_var='''
Despite being almost evenly distributed across marketing channels,
personalized ads successfully encouraged more users to subscribe
compared to standard (control) ads.'''

plt.text(2,.1,text_con_var,color='#313E4C')
plt.text(3,.06,text2_con_var, ha='center',fontstyle='italic',weight='semibold', fon

plt.show()


# 2-Visualization - Retention Rates Across Variant Classifications:
bars(retention_var,'Ad Classification','Retention Rate',retention_rate)

# Additional Customization
plt.title('Retention Rates Across Ad Classifications')

plt.yticks(np.arange(0,.9,.1),[f'{y:.0%}' for y in np.arange(0,.9,.1)], fontsize=8,

# Findings
text_ret_var=f'''
- Control ads achieved a higher Retention Rate ({retention_var['Retention Rate'].ma
  compared to the Pesonalized group ({retention_var['Retention Rate'].min():.2%}).\
- Retention Rate of the Controlled Ads is more than the Overall
  Retention Rate of {retention_rate:.2%}, while the personalized ads were
  relatively colse to it'''

text2_ret_var='''
Users acquired through Controlled ads are more likely to stay
```

```
subscribed'''

plt.text(2,.4,text_ret_var,color='#313E4C')
plt.text(3,.2,text2_ret_var, ha='center',fontstyle='italic',weight='semibold', font

plt.show();
```



- Personalized ads achieved a higher conversion rate (16.80%) than both the control group (9.00%) and the overall conversion rate (13.01%).

Despite being almost evenly distributed across marketing channels, personalized ads successfully encouraged more users to subscribe compared to standard (control) ads.



- Control ads achieved a higher Retention Rate (67.63%) compared to the Pesonalized group (65.50%).

- Retention Rate of the Controlled Ads is more than the Overall Retention Rate of 65.95%, while the personalized ads were relatively colse to it

Users acquired through Controlled ads are more likely to stay subscribed

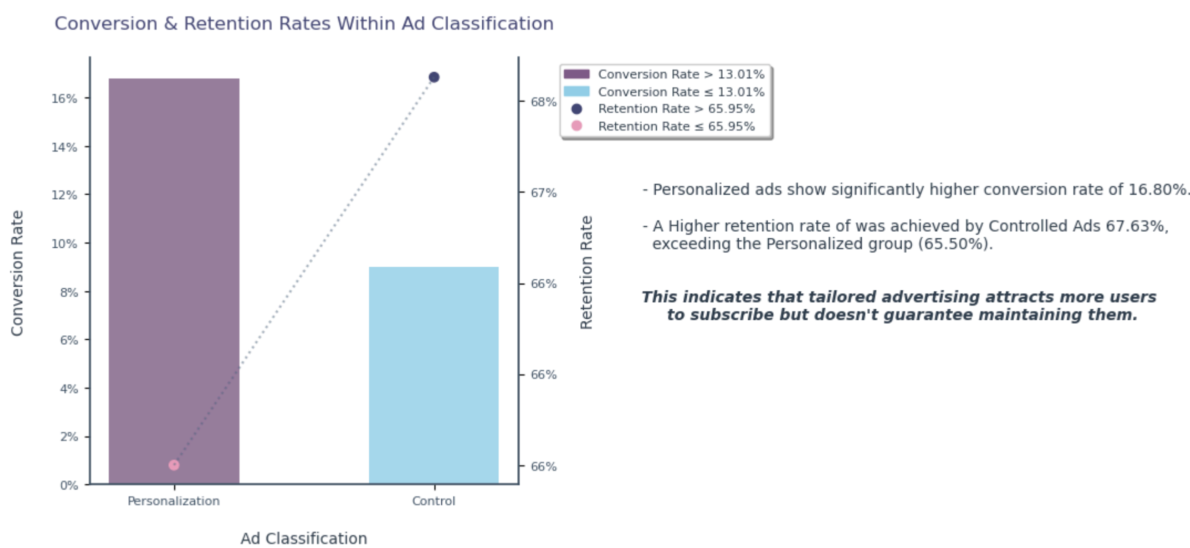
In [539…
```
# Comparing Conversion Rates with Retention Rates Within Variant Classification:
performance_var=comparison(marketing,converted_users,retained_users,['variant'],'us

performance_var.columns=['Ad Classification','Conversion Rate','Retention Rate']

performance_var.style.hide().format({'Conversion Rate':'{:,.2%}','Retention Rate':'
```

Out[539…

Ad Classification	Conversion Rate	Retention Rate
personalization	16.80%	65.50%
control	9.00%	67.63%

In [540…
```
# Visualization - Comparing Conversion Rates with Retention Rates Within Variant Cl
combo(performance_var,'Ad Classification','Conversion Rate','Retention Rate', conve

# Additional Customization
```

```
plt.title('\nConversion & Retention Rates Within Ad Classification\n', fontsize=12,

# Findings
text_perform_var=f'''
- Personalized ads show significantly higher conversion rate of {performance_var['C
- A Higher retention rate of was achieved by Controlled Ads {performance_var['Reten
  exceeding the Personalized group ({performance_var['Retention Rate'].min():.2%}).

text2_perform_var='''
This indicates that tailored advertising attracts more users
to subscribe but doesn't guarantee maintaining them.'''

plt.text(1.8,.6669,text_perform_var, color='#313E4C')
plt.text(2.8,.663,text2_perform_var, ha='center',fontstyle='italic',weight='semibol

plt.show();
```



3 – *Displayed Language:*

In [541…

```
# Calculating Conversion Rate across Displayed Languages
conversion_displayed_lang=con_ret(marketing,converted_users,'language_displayed','u

conversion_displayed_lang.columns=['Language Displayed', 'Total Users','Converted',

conversion_displayed_lang.style.hide().format({'Conversion Rate':'{:,.2%}'})
```

Out[541…

Language Displayed	Total Users	Converted	Conversion Rate
German	73	53	72.60%
Arabic	24	12	50.00%
Spanish	120	24	20.00%
English	7720	936	12.12%

In [542…

```
# Calculating Retention Rate across Displayed Languages
retention_displayed_lang=con_ret(converted_users,retained_users,'language_displayed

retention_displayed_lang.columns=['Language Displayed','Converted','Retained','Rete

retention_displayed_lang.style.hide().format({'Retention Rate':'{:,.2%}'})
```

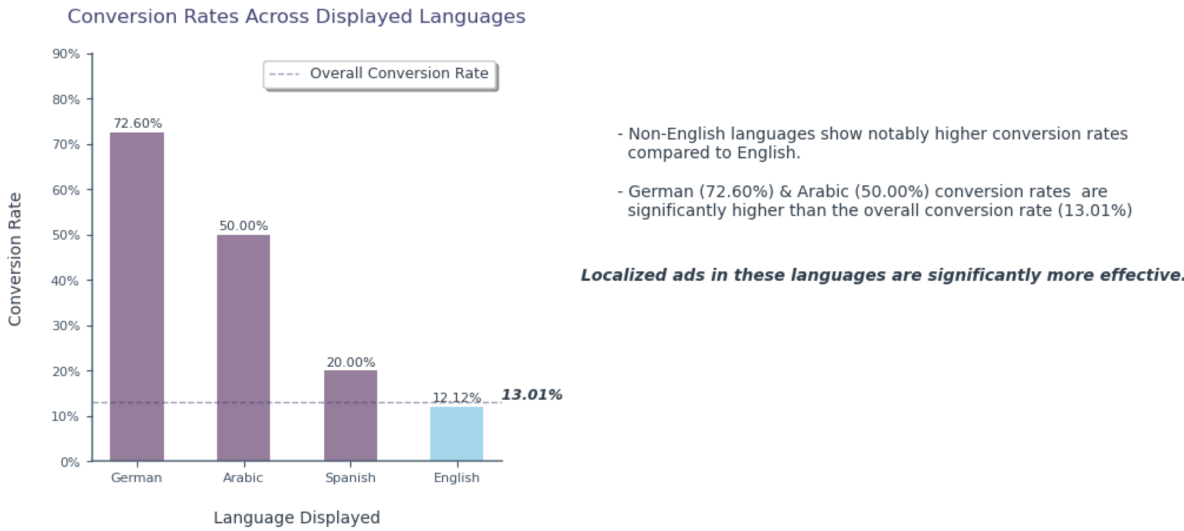
Out[542…

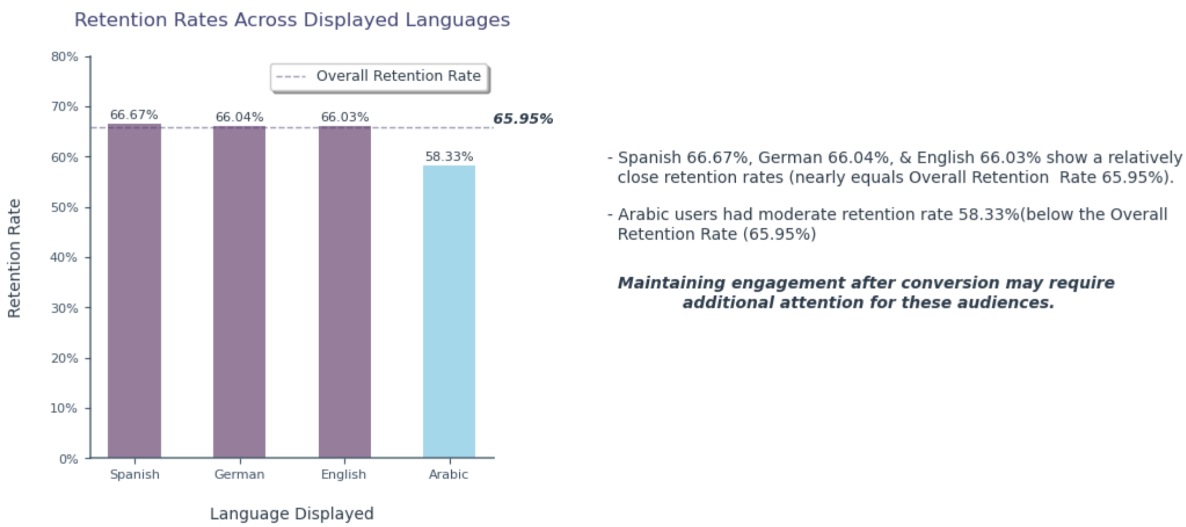
Language Displayed	Converted	Retained	Retention Rate
Spanish	24	16	66.67%
German	53	35	66.04%
English	936	618	66.03%
Arabic	12	7	58.33%

In [543…

```
# 1- Visualization - Conversion Rates Across Displayed Languages:
bars(conversion_displayed_lang,'Language Displayed','Conversion Rate',conversion_ra

# Additional Customization
plt.title('\nConversion Rates Across Displayed Languages\n')
plt.yticks(np.arange(0,1,.1),[f'{y:.0%}' for y in np.arange(0,1,.1)])

# Findings
text_con_dlang=f'''
- Non-English languages show notably higher conversion rates
  compared to English.\n
- German ({conversion_displayed_lang.iloc[0,3]:.2%}) & Arabic ({conversion_displaye
  significantly higher than the overall conversion rate ({conversion_rate:.2%})
'''
text2_con_dlang='''
Localized ads in these languages are significantly more effective.'''

plt.text(4.5,.5,text_con_dlang,color='#313E4C')
plt.text(7,.4,text2_con_dlang, ha='center',fontstyle='italic',weight='semibold', fo

plt.show()


# 2- Visualization - Retention Rates Across Displayed Languages:
bars(retention_displayed_lang,'Language Displayed','Retention Rate',retention_rate)

# Additional Customization
plt.title('\nRetention Rates Across Displayed Languages\n')
plt.yticks(np.arange(0,.9,.1),[f'{y:.0%}' for y in np.arange(0,.9,.1)])

# Findings
text_ret_dlang=f'''
- {retention_displayed_lang.iloc[0,0]} {retention_displayed_lang.iloc[0,3]:.2%}, {r
  close retention rates (nearly equals Overall Retention  Rate {retention_rate:.2%}
- {retention_displayed_lang.iloc[3,0]} users had moderate retention rate {retention
  Retention Rate ({retention_rate:.2%})
 '''
text2_ret_dlang='''
```

```
Maintaining engagement after conversion may require
additional attention for these audiences.'''

plt.text(4.5,.4,text_ret_dlang,color='#313E4C')
plt.text(7,.3,text2_ret_dlang, ha='center',fontstyle='italic',weight='semibold', fo

plt.show()
```





In [544…

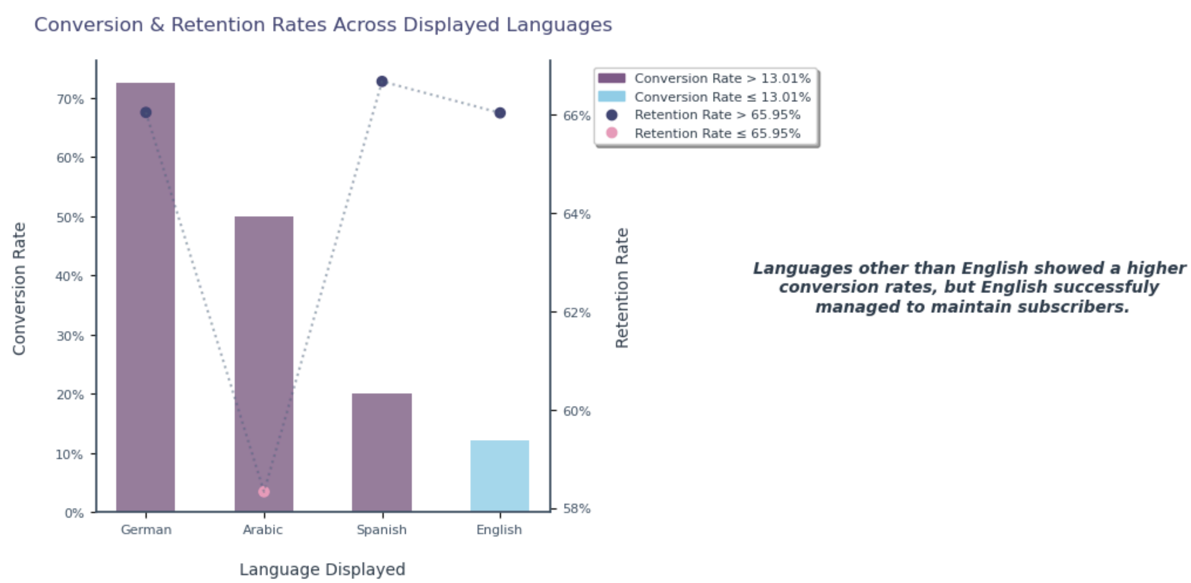
```
# Comparing Conversion Rates with Retention Rates Across Displayed Languages:
performance_displayed_lang=comparison(marketing,converted_users,retained_users,['la

performance_displayed_lang.columns=['Language Displayed','Conversion Rate','Retenti

performance_displayed_lang.style.hide().format({'Conversion Rate':'{:,.2%}','Retent
```

Out[544...

Language Displayed	Conversion Rate	Retention Rate
German	72.60%	66.04%
Arabic	50.00%	58.33%
Spanish	20.00%	66.67%
English	12.12%	66.03%

In [545...

```
# Visualization - Comparing Conversion Rates with Retention Rates Across Displayed
combo(performance_displayed_lang,'Language Displayed','Conversion Rate','Retention

# Additional Customization
plt.title('\nConversion & Retention Rates Across Displayed Languages\n')

# Findings
text_perform_dlang='''
Languages other than English showed a higher
conversion rates, but English successfuly
managed to maintain subscribers.'''
plt.text(7,.62,text_perform_dlang, ha='center',fontstyle='italic',weight='semibold'
plt.show();
```



4 – *Matched Language:*

In [546...

```
# Calculating Conversion Rate across Matched Languages
conversion_lang=con_ret(marketing,converted_users,'matched_lang','user_id')

conversion_lang.columns=['Language Status', 'Total Users', 'Converted', 'Conversion

conversion_lang['Language Status'] = np.where (conversion_lang['Language Status']==

conversion_lang.style.hide().format({'Conversion Rate':'{:,.2%}'})
```

Out[546…

Language Status	Total Users	Converted	Conversion Rate
Matched	7531	998	13.25%
Not_Matched	403	27	6.70%

In [547…

```
# Calculating Retention Rate across Matched & Not-Matched Languages
retention_lang=con_ret(converted_users,retained_users,'matched_lang','user_id')

retention_lang.columns=['Language Status', 'Converted','Retained', 'Retention Rate'

retention_lang['Language Status'] = np.where (retention_lang['Language Status']==Tr

retention_lang.style.hide().format({'Retention Rate':'{:,.2%}'})
```

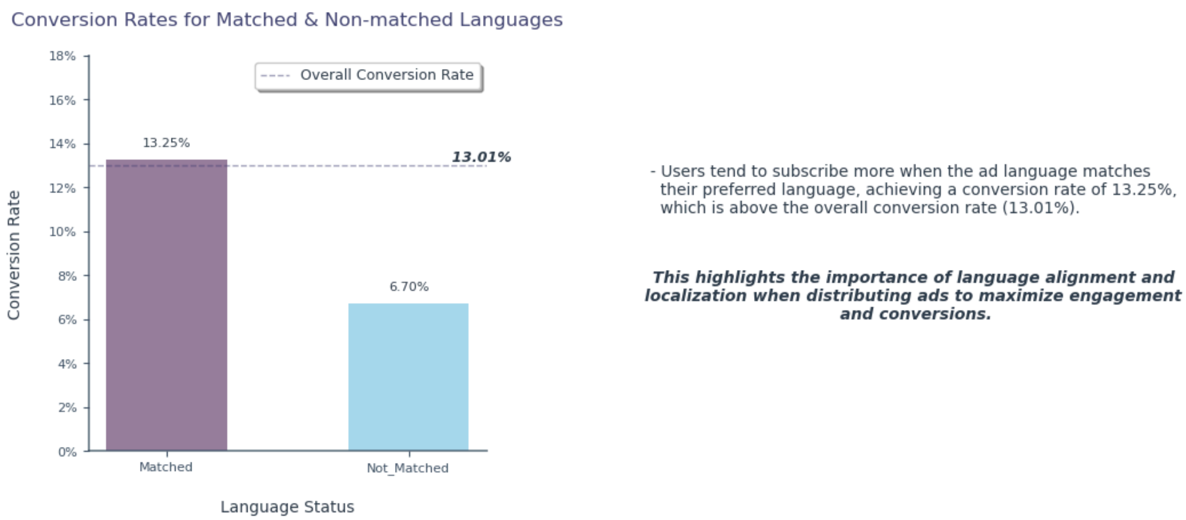
Out[547…

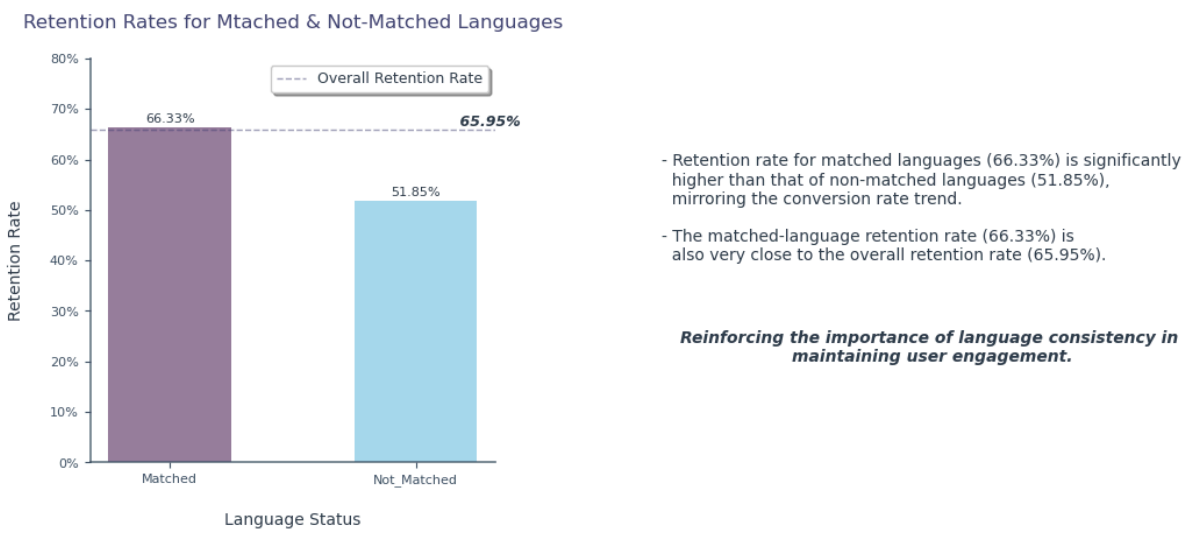
Language Status	Converted	Retained	Retention Rate
Matched	998	662	66.33%
Not_Matched	27	14	51.85%

In [548…

```
# 1- Visualization - Conversion Rates for Matched & Non-matched Languages:
bars(conversion_lang,'Language Status','Conversion Rate',conversion_rate)

#Additional Customization
plt.title('\nConversion Rates for Matched & Non-matched Languages\n')

plt.yticks(np.arange(0,.2,.02),[f'{y:.0%}' for y in np.arange(0,.2,.02)])


# Findings
text_con_lstatus=f'''
- Users tend to subscribe more when the ad language matches
  their preferred language, achieving a conversion rate of {conversion_lang.iloc[0,
  which is above the overall conversion rate ({conversion_rate:.2%}).\n'''

text2_con_lstatus='''
This highlights the importance of language alignment and
localization when distributing ads to maximize engagement
and conversions.'''

plt.text(2,.1,text_con_lstatus,color='#313E4C')
plt.text(3.1,.06,text2_con_lstatus, ha='center',fontstyle='italic',weight='semibold

plt.show()


# 2- Visualization - Retention Rates for Mtached & Not-Matched Languages:
bars(retention_lang,'Language Status','Retention Rate',retention_rate)

# Additional Customization
plt.title('\nRetention Rates for Mtached & Not-Matched Languages\n')

plt.yticks(np.arange(0,.9,.1),[f'{y:.0%}' for y in np.arange(0,.9,.1)])
```

```
# Findings
text_ret_lstatus=f'''
- Retention rate for matched languages ({retention_lang.iloc[0,3]:.2%}) is signific
  higher than that of non-matched languages ({retention_lang.iloc[1,3]:.2%}),
  mirroring the conversion rate trend.\n
- The matched-language retention rate ({retention_lang.iloc[0,3]:.2%}) is
  also very close to the overall retention rate ({retention_rate:.2%}).'''

text2_ret_lstatus='''
Reinforcing the importance of language consistency in
maintaining user engagement.'''

plt.text(2,.4,text_ret_lstatus,color='#313E4C')
plt.text(3.1,.2,text2_ret_lstatus, ha='center',fontstyle='italic',weight='semibold'

plt.show()
```





In [549…

```
# Comparing Conversion Rates with Retention Rates Within Matched & Not-Matched Lang
performance_lang=comparison(marketing,converted_users,retained_users,['matched_lang

performance_lang.columns=['Language Status','Conversion Rate','Retention Rate']
```

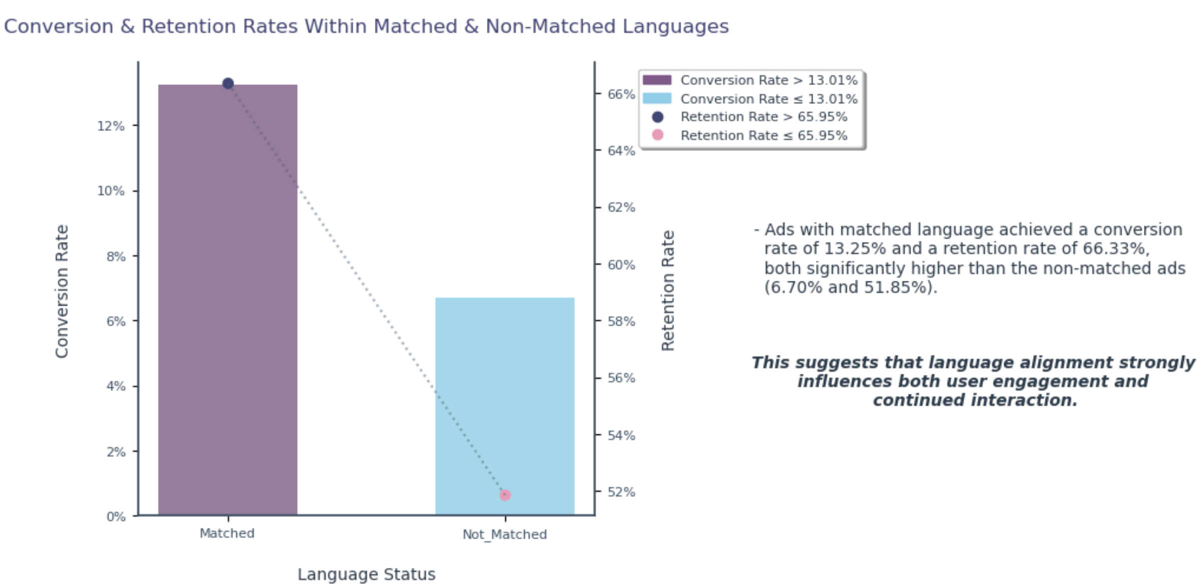
```
performance_lang['Language Status'] = np.where(performance_lang['Language Status']=

performance_lang.style.hide().format({'Conversion Rate':'{:,.2%}','Retention Rate':
```

Out[549…

Language Status	Conversion Rate	Retention Rate
Matched	13.25%	66.33%
Not_Matched	6.70%	51.85%

In [550…

```
# Visualization - Comparing Conversion Rates with Retention Rates Within Matched &
combo(performance_lang,'Language Status','Conversion Rate','Retention Rate', conver

# Additional Customization
plt.title('\nConversion & Retention Rates Within Matched & Non-Matched Languages\n'

# Findings
text_perform_lstatus=f'''
- Ads with matched language achieved a conversion
  rate of {performance_lang.iloc[0,1]:.2%} and a retention rate of {performance_lan
  both significantly higher than the non-matched ads
  ({performance_lang.iloc[1,1]:.2%} and {performance_lang.iloc[1,2]:.2%}).'''

text2_perform_lstatus='''
This suggests that language alignment strongly
influences both user engagement and
continued interaction.'''

plt.text(1.9,.59,text_perform_lstatus, color='#313E4C')
plt.text(2.7,.55,text2_perform_lstatus, ha='center',fontstyle='italic',weight='semi

plt.show();
```



> *Note*:
> *The huge gap between the number of users with matched language (7531 user) and the nit matched language (403 users) should be taken into consideration when further investigating these noticable gaps in conversion and retention rates.*

5 – *Age Groups:*

In [551…

```
# Calculating Conversion Rate within Age Groups:
conversion_age =con_ret(marketing,converted_users,'age_group','user_id').sort_value

conversion_age.columns=['Age Group', 'Total Users', 'Converted', 'Conversion Rate']

conversion_age['Age Group'] =conversion_age['Age Group'].apply(lambda x: x.replace(

conversion_age.style.hide().format({'Conversion Rate':'{:,.2%}'})
```

Out[551…

Age Group	Total Users	Converted	Conversion Rate
0-18	1206	192	15.92%
19-24	1304	303	23.24%
24-30	1218	228	18.72%
30-36	1057	77	7.28%
36-45	1056	74	7.01%
45-55	1056	75	7.10%
55+	979	76	7.76%

In [552…

```
# Calculating Retention Rate within Age Groups:
retention_age= con_ret(converted_users,retained_users,'age_group','user_id').sort_v

retention_age.columns=['Age Group', 'Converted', 'Retained','Retention Rate']

retention_age['Age Group'] =retention_age['Age Group'].apply(lambda x: x.replace('

retention_age.style.hide().format({'Retention Rate':'{:,.2%}'})
```

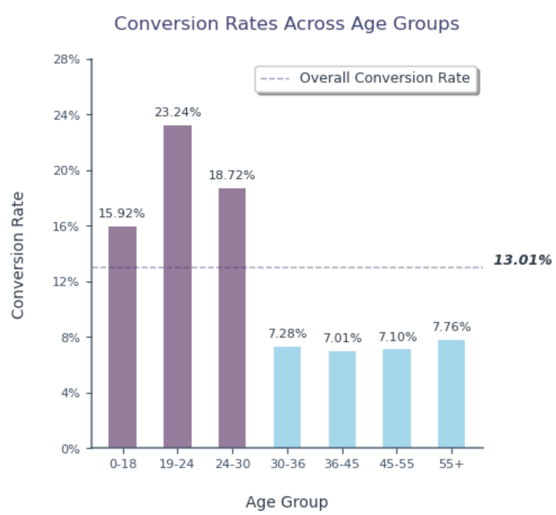
Out[552...

Age Group	Converted	Retained	Retention Rate
0-18	192	126	65.62%
19-24	303	208	68.65%
24-30	228	150	65.79%
30-36	77	52	67.53%
36-45	74	45	60.81%
45-55	75	45	60.00%
55+	76	50	65.79%

In [553...

```
# 1- Visualization - Conversion Rates Across Age Groups:
bars(conversion_age,'Age Group','Conversion Rate',conversion_rate)

# Additional Customization
plt.title('\nConversion Rates Across Age Groups\n')

plt.yticks(np.arange(0,.32,.04),[f'{y:.0%}' for y in np.arange(0,.32,.04)])

# Findings
text_con_age=f'''
Younger Users (Under 30) → \n
  - 19–24 years achieved the highest conversion rate ({conversion_age['Conversion R
    followed by 24–30 years ({sorted(conversion_age['Conversion Rate'], reverse=Tru
  - This is consistent with the Ad Distribution across Age Groups.\n
Older Users (Above 30) → \n
  - showed a significant drop in engagement, with conversion rates
    around {np.mean(sorted(conversion_age['Conversion Rate'])[0:4]):.2%}. \n
  - Are less likely to engage or convert, suggesting that ad content
    or platform selection may not align well with their preferences.\n'''

text2_con_age='''
It may be valuable to tailor messaging or channels for older
age segments while maintaining strong targeting toward younger
audiences who show higher conversion potential.'''

plt.text(9.5,.045,text_con_age, color='#313E4C')
plt.text(14,0.005,text2_con_age, ha='center',fontstyle='italic',weight='semibold',

plt.show()

# 2- Visualization - Retention Rates Across Age Groups:
bars(retention_age,'Age Group','Retention Rate',retention_rate)

# Additional Customization
plt.title('\nRetention Rates Across Age Groups\n')

plt.yticks(np.arange(0,1,.1),[f'{y:.0%}' for y in np.arange(0,1,.1)])

# Findings
text3=f'''
```

```
Ages 19-24 & 30-36 → \n
  - Form the highest retention rates (≈ {np.mean(sorted(retention_age['Retention Ra
Ages 0-18, 24-30, & 55+ → \n
  - Despite being lower than the overall retention rate ({retention_rate:.2%}),
    they are relatively colse to it. \n\n
Ages 30-36 & 55+ → \n
  - Tend to retain in a moderate rate despite their lower
    conversion rates'''
text4='''
Ages 30-36 & above 55, While less likely to convert,
exhibit better loyalty once they do.'''
plt.text(9.5,.2,text3, color='#313E4C')
plt.text(14,.05,text4, ha='center',fontstyle='italic',weight='semibold', fontsize=1

plt.show()
```



Younger Users (Under 30) →

- 19–24 years achieved the highest conversion rate (23.24%), followed by 24–30 years (18.72%) and 0–18 years (15.92%)

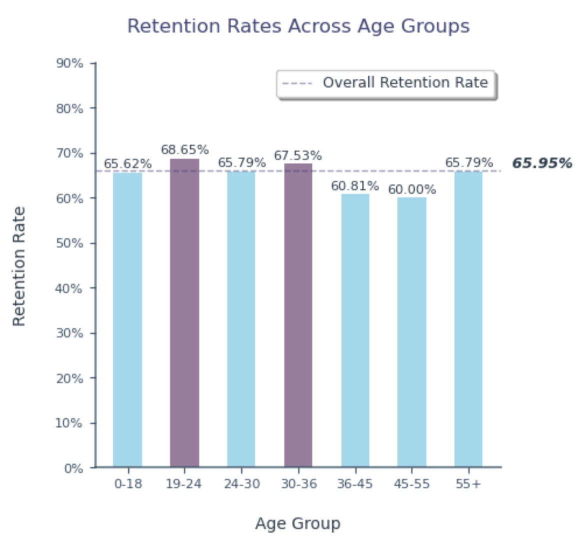
- This is consistent with the Ad Distribution across Age Groups.

Older Users (Above 30) →

- showed a significant drop in engagement, with conversion rates around 7.29%.

- Are less likely to engage or convert, suggesting that ad content or platform selection may not align well with their preferences.

It may be valuable to tailor messaging or channels for older age segments while maintaining strong targeting toward younger audiences who show higher conversion potential.



Ages 19-24 & 30-36 →

- Form the highest retention rates (≈ 68%).

Ages 0-18, 24-30, & 55+ →

- Despite being lower than the overall retention rate (65.95%), they are relatively colse to it.

Ages 30-36 & 55+ →

- Tend to retain in a moderate rate despite their lower conversion rates

Ages 30-36 & above 55, While less likely to convert, exhibit better loyalty once they do.

In [554…

```
# Comparing Conversion Rates with Retention Rates within Age Groups:
performance_age=performance_lang=comparison(marketing,converted_users,retained_user

performance_age['Age Group']=performance_age['Age Group'].apply(lambda x: x.replace
```

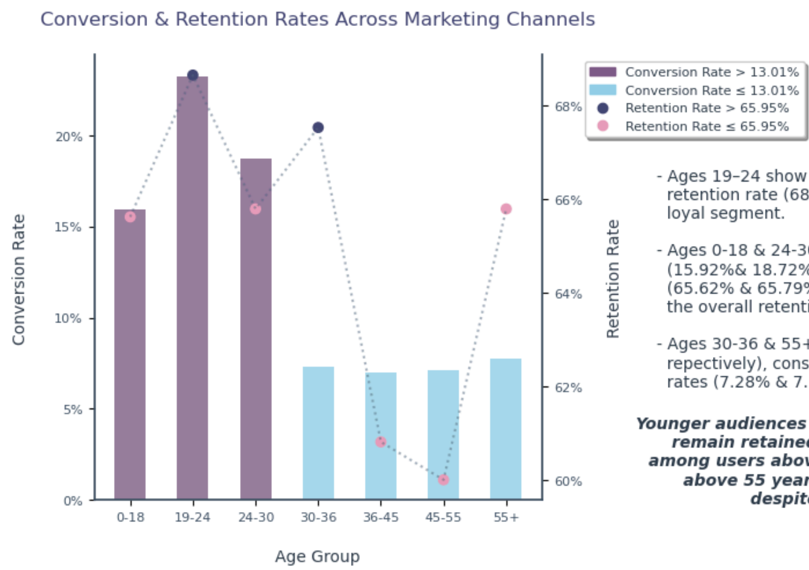
```
performance_age.style.hide().format({'Conversion Rate':'{:,.2%}','Retention Rate':'
```

Out[554…

Age Group	Conversion Rate	Retention Rate
0-18	15.92%	65.62%
19-24	23.24%	68.65%
24-30	18.72%	65.79%
30-36	7.28%	67.53%
36-45	7.01%	60.81%
45-55	7.10%	60.00%
55+	7.76%	65.79%

In [555…

```
# Visualization - Comparing Conversion Rates with Retention Rates within Age Groups
combo(performance_age,'Age Group','Conversion Rate','Retention Rate', conversion_ra

# Additional Customization
plt.title('\nConversion & Retention Rates Across Marketing Channels\n')
# Findings
text_perform_age=f'''
- Ages 19-24 show the highest conversion rate ({performance_age['Conversion Rate'].
  retention rate ({performance_age['Retention Rate'].max():.2%}), making this the m
  loyal segment.\n
- Ages 0-18 & 24-30 showed a considerably high conversion rates
  ({sorted(conversion_age['Conversion Rate'], reverse=True)[2]:.2%}& {sorted(conver
  ({performance_age[performance_age['Age Group']=='0-18'].iloc[0,2]:.2%} & {perform
  the overall retention rate ({retention_rate:.2%}.\n
- Ages 30-36 & 55+ showed more loyality ({performance_age[performance_age['Age Grou
  repectively), considering their significantly low conversion
  rates ({performance_age[performance_age['Age Group']=='30-36'].iloc[0,1]:.2%} & {

text2_perform_age='''
Younger audiences (under 30) are more likely to engage and
remain retained after conversion, while engagement
among users above 30 is weaker. Except for ages 30-36 &
above 55 years who tend to retain in a high range
despite their lower conversion rate.'''

plt.text(8.4,.62,text_perform_age, color='#313E4C')
plt.text(12.5,.595,text2_perform_age, ha='center',fontstyle='italic',weight='semibo

plt.show();
```



- Ages 19–24 show the highest conversion rate (23.24%) and retention rate (68.65%), making this the most responsive and loyal segment.

- Ages 0-18 & 24-30 showed a considerably high conversion rates (15.92%& 18.72%respectively) with a moderate retention rates (65.62% & 65.79% repectively) that are slightly lower than the overall retention rate (65.95%.

- Ages 30-36 & 55+ showed more loyality (67.53% & 65.79% repectively), considering their significantly low conversion rates (7.28% & 7.76% repectively)

Younger audiences (under 30) are more likely to engage and remain retained after conversion, while engagement among users above 30 is weaker. Except for ages 30-36 & above 55 years who tend to retain in a high range despite their lower conversion rate.

6 – *Date Served:*

In [556…

```
# Calculating Conversion Rate Within Served Dates
conversion_date = con_ret(marketing,converted_users,'date_served', 'user_id').sort_

conversion_date.columns=['Date Served', 'Total Users', 'Converted', 'Conversion Rat

conversion_date.style.hide().format({'Date Served': lambda x: x.strftime('%Y-%m-%d'
```

Out[556…

Date Served	Total Users	Converted	Conversion Rate
2018-01-01	363	36	9.92%
2018-01-02	378	37	9.79%
2018-01-03	354	36	10.17%
2018-01-04	330	35	10.61%
2018-01-05	325	40	12.31%
2018-01-06	312	35	11.22%
2018-01-07	277	39	14.08%
2018-01-08	314	36	11.46%
2018-01-09	313	39	12.46%
2018-01-10	337	40	11.87%
2018-01-11	311	25	8.04%
2018-01-12	301	23	7.64%
2018-01-13	306	26	8.50%
2018-01-14	305	26	8.52%
2018-01-15	778	87	11.18%
2018-01-16	388	99	25.52%
2018-01-17	369	81	21.95%
2018-01-18	318	29	9.12%
2018-01-19	305	18	5.90%
2018-01-20	311	21	6.75%
2018-01-21	229	20	8.73%
2018-01-22	180	22	12.22%
2018-01-23	175	21	12.00%
2018-01-24	192	22	11.46%
2018-01-25	184	23	12.50%
2018-01-26	222	20	9.01%
2018-01-27	321	21	6.54%
2018-01-28	320	20	6.25%
2018-01-29	319	19	5.96%
2018-01-30	318	21	6.60%

Date Served	Total Users	Converted	Conversion Rate
2018-01-31	341	18	5.28%

In [557…

```
# Calculating Retention Rate Within Served Dates
retention_date = con_ret(converted_users,retained_users,'date_served', 'user_id').s

retention_date.columns=['Date Served', 'Total Users', 'Converted', 'Retention Rate'

retention_date.style.hide().format({'Date Served': lambda x: x.strftime('%Y-%m-%d')
```

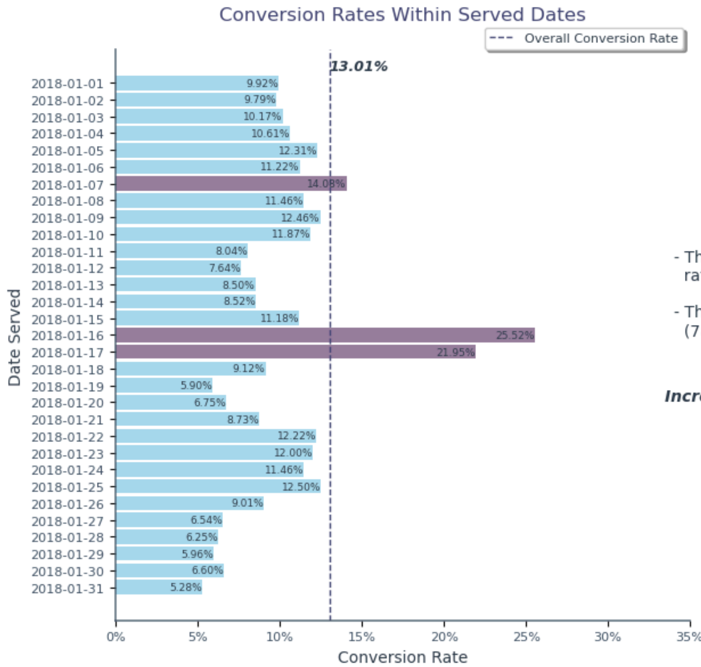
Out[557…

Date Served	Total Users	Converted	Retention Rate
2018-01-01	36	28	77.78%
2018-01-02	37	26	70.27%
2018-01-03	36	27	75.00%
2018-01-04	35	17	48.57%
2018-01-05	40	23	57.50%
2018-01-06	35	26	74.29%
2018-01-07	39	20	51.28%
2018-01-08	36	22	61.11%
2018-01-09	39	25	64.10%
2018-01-10	40	26	65.00%
2018-01-11	25	14	56.00%
2018-01-12	23	8	34.78%
2018-01-13	26	13	50.00%
2018-01-14	26	16	61.54%
2018-01-15	87	70	80.46%
2018-01-16	99	68	68.69%
2018-01-17	81	49	60.49%
2018-01-18	29	21	72.41%
2018-01-19	18	12	66.67%
2018-01-20	21	17	80.95%
2018-01-21	20	15	75.00%
2018-01-22	22	17	77.27%
2018-01-23	21	15	71.43%
2018-01-24	22	15	68.18%
2018-01-25	23	13	56.52%
2018-01-26	20	16	80.00%
2018-01-27	21	12	57.14%
2018-01-28	20	14	70.00%
2018-01-29	19	14	73.68%
2018-01-30	21	15	71.43%

Date Served	Total Users	Converted	Retention Rate
2018-01-31	18	12	66.67%

In [558…

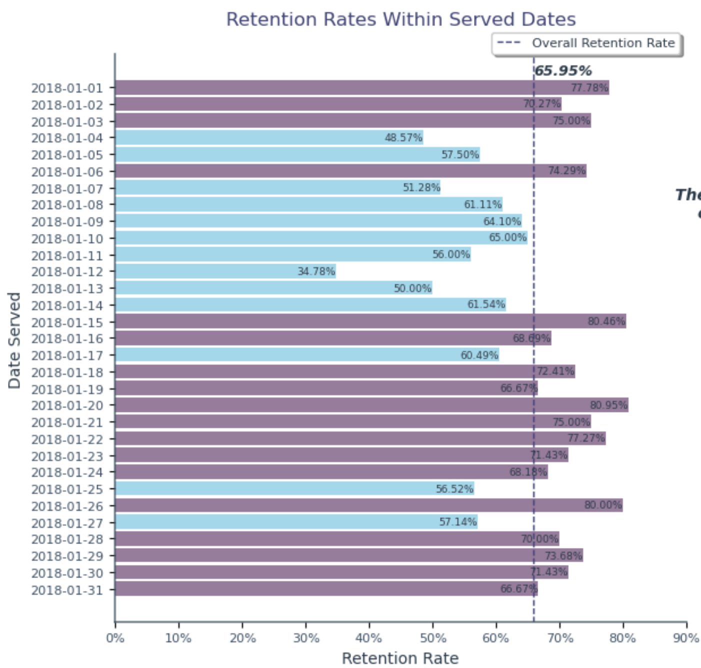
```
# 1- Visualization - Conversion Rates Within Served Dates
h_bar(conversion_date,'Date Served', 'Conversion Rate', conversion_rate)

# Additional Customization
plt.title('\nConversion Rates Within Served Dates\n')
plt.xticks(np.arange(0,.40,.05),[f'{x:.0%}' for x in np.arange(0,.4,.05)])

# Findings
text_con_sdate=f'''
- The 16th and 17th of January recorded the highest conversion
  rates at {conversion_date['Conversion Rate'].to_list()[15]:.2%} & {conversion_dat
- This spike aligns with the peak in ad views on January 15
  ({daily_users['Daily Users'].max()} view).'''

text2_con_sdate='''
Increased exposure and campaign intensity could positively
influence user conversions over the following days.'''

plt.text(.34,15,text_con_sdate, color='#313E4C')
plt.text(.5,20,text2_con_sdate, ha='center',fontstyle='italic',weight='semibold', f

plt.show()

# 2- Visualization - Retention Rates Within Served Dates:
h_bar(retention_date,'Date Served', 'Retention Rate', retention_rate)

# Additional Customization
plt.title('\nRetention Rates Within Served Dates\n')

plt.xticks(np.arange(0,1,.1),[f'{x:.0%}' for x in np.arange(0,1,.1)])

# Findings
text_ret_sdate='''
The retention rates across different dates show no clear or
consistent pattern, indicating that user retention was
not strongly influenced by the specific date
the ad was served.'''

plt.text(1.3,10,text_ret_sdate, ha='center',fontstyle='italic',weight='semibold', f

plt.show()
```



- The 16th and 17th of January recorded the highest conversion rates at 25.52% & 21.95%, respectively.

- This spike aligns with the peak in ad views on January 15 (784 view).

Increased exposure and campaign intensity could positively influence user conversions over the following days.



The retention rates across different dates show no clear or consistent pattern, indicating that user retention was not strongly influenced by the specific date the ad was served.

$Q2:$ *Is there evidence that multi-touch exposure (users seeing multiple ads) improves conversion or retention rates?*

In [559…

```
# Calculating Conversion Rate across single & multi-touch exposure

conversion_exposure =con_ret(marketing,converted_users,'ad_repeated','user_id')

conversion_exposure.columns=['Ad Exposure','Total Users','Converted', 'Conversion R

conversion_exposure['Ad Exposure']= np.where(conversion_exposure['Ad Exposure']==Tr
```

```
conversion_exposure.style.hide().format({'Conversion Rate':"{:,.2%}"})
```

Out[559…

Ad Exposure	Total Users	Converted	Conversion Rate
Multi_Exposure	1806	322	17.83%
Single_Exposure	6070	703	11.58%

In [560…

```
# Calculating Retention Rate across single & multi-touch exposure
retention_exposure =con_ret(converted_users,retained_users,'ad_repeated','user_id')

retention_exposure.columns=['Ad Exposure','Total Users','Retained', 'Retention Rate

retention_exposure['Ad Exposure']= np.where(retention_exposure['Ad Exposure']==True

retention_exposure.style.hide().format({'Retention Rate':"{:,.2%}"})
```

Out[560…

Ad Exposure	Total Users	Retained	Retention Rate
Multi_Exposure	322	226	70.19%
Single_Exposure	703	450	64.01%

In [561…

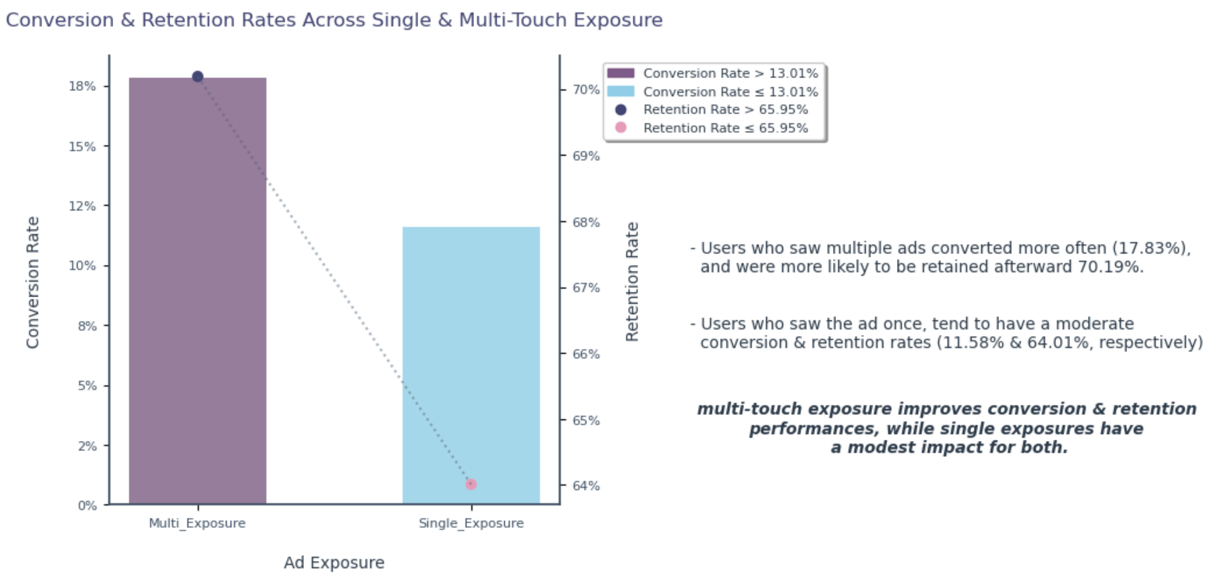
```
# merging conversion & retention dates across single & multi-touch exposure
performance_exposure=conversion_exposure.merge(retention_exposure,on='Ad Exposure')
performance_exposure.style.hide().format({'Conversion Rate':'{:,.2%}','Retention Ra
```

Out[561…

Ad Exposure	Conversion Rate	Retention Rate
Multi_Exposure	17.83%	70.19%
Single_Exposure	11.58%	64.01%

In [562…

```
# Visualization - Conversion & Retention Rates Across Single & Multi-Touch Exposure
combo(performance_exposure,'Ad Exposure','Conversion Rate',      'Retention Rate',co
plt.title('\nConversion & Retention Rates Across Single & Multi-Touch Exposure\n')

# Findings
text_perform_exposure=f'''
- Users who saw multiple ads converted more often ({performance_exposure['Conversio
  and were more likely to be retained afterward {performance_exposure['Retention Ra

- Users who saw the ad once, tend to have a moderate
  conversion & retention rates ({performance_exposure['Conversion Rate'].min():.2%}
'''

text2_perform_exposure='''
multi-touch exposure improves conversion & retention
performances, while single exposures have
a modest impact for both.'''

plt.text(1.8,.658,text_perform_exposure, color='#313E4C', ha='left')
plt.text(2.75,.645,text2_perform_exposure, ha='center',fontstyle='italic',weight='s
plt.show();
```



> *Note*:
> *Around 70% of unique users (~5,000) were exposed to the ad only once. This large proportion could be aligned with the extensive ad distribution through House Ads, which had the lowest conversion and retention rates (7.51% & 58.05%, respectively).*

In [563...

```
# Single & Multi-Exposures Across Marketing Channels:
users_exposure_house = marketing.groupby(['ad_repeated','is_house_ad']).user_id.nun
                                .rename(columns={'ad_repeated':'Ad Exposure','is_ho

users_exposure_house['Ad Exposure']=np.where(users_exposure_house['Ad Exposure']==T

users_exposure_house['Ad Type']=np.where(users_exposure_house['Ad Type']==True,'Hou

users_exposure_house =users_exposure_house.pivot_table(values='total_users', index=
                                          .apply(lambda x: x/sum(x)).reset_index()

users_exposure_house.style.hide().format({'Multi_Exposure':'{:,.2%}','Single_Exposu
```

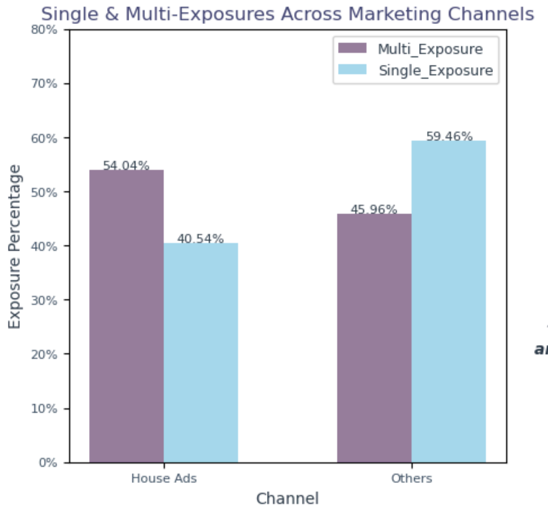
Out[563...

Ad Type	Multi_Exposure	Single_Exposure
House Ad	54.04%	40.54%
Others	45.96%	59.46%

In [564...

```
# Visualization - Single & Multi-Exposures Across Marketing Channels:
#Data:
x=np.arange(len(users_exposure_house['Ad Type']))
y=users_exposure_house['Multi_Exposure']
z=users_exposure_house['Single_Exposure']

# Creating the Chart:
plt.subplots(figsize=(5,5))
```

```python
width=.3
location=x+width/2

# 1- Multi-Exposure:
plt.bar(x,y,width, label='Multi_Exposure',color='#805D87', alpha=.8)

# 2- Single-Exposure:
plt.bar(x+width,z,width,label='Single_Exposure',color='#94D1E7',alpha=.8)

# Customizing the Chart:
plt.title('Single & Multi-Exposures Across Marketing Channels', fontsize=12, color=

plt.xlabel('Channel', fontsize=10, color='#313E4c')
plt.xticks(location, ['House Ads', 'Others'], fontsize=8, color='#415366')

plt.ylabel('Exposure Percentage', fontsize=10, color='#313E4c')
plt.yticks(np.arange(0,.9,.1), [f'{y:.0%}' for y in np.arange(0,.9,.1)], fontsize=8

plt.legend(fontsize=9,labelcolor='#313E4C',loc='upper right')

# Annotating Values to the Chart:
for i,v in enumerate(y):
    plt.text(i,v,f'{v:,.2%}', ha='center', fontsize=8, color='#313E4c')

for r,s in enumerate(z):
    plt.text(r+width,s,f'{s:,.2%}', ha='center', fontsize=8, color='#313E4c')

# Findings:
text1= f'''
- House Ads make up {z.to_list()[0]:.2%} of single-exposures, a sizable
  share compared to other marketing channels ({z.to_list()[1]:.2%}).\n
- {y.to_list()[0]:.2%} of mutli-exposures originate from House Ads indicating an al
  dominant presence compared to the {y.to_list()[1]:.2%} from other marketing chann
text2=f'''
House Ads likely drive the high number of single-exposure users,
showing a strong influence compared to other marketing channels.
This imbalance may help explain the noticeably low conversion ({performance_exposur
and retention ({performance_exposure['Retention Rate'].min():.2%}) rates observed w
plt.text(1.8,.4,text1, color='#313E4C', ha='left')
plt.text(3,.2,text2, ha='center',fontstyle='italic',weight='semibold', fontsize=10,

plt.show();
```



- House Ads make up 40.54% of single-exposures, a sizable share compared to other marketing channels (59.46%).

- 54.04% of mutli-exposures originate from House Ads indicating an almost dominant presence compared to the 45.96% from other marketing channels.

House Ads likely drive the high number of single-exposure users, showing a strong influence compared to other marketing channels. This imbalance may help explain the noticeably low conversion (11.58%) and retention (64.01%) rates observed within the single-exposure category.

Audience and Channel Interaction :

*Q*3 : ***Which combinations of age group and marketing channel yield the highest conversion rates?***

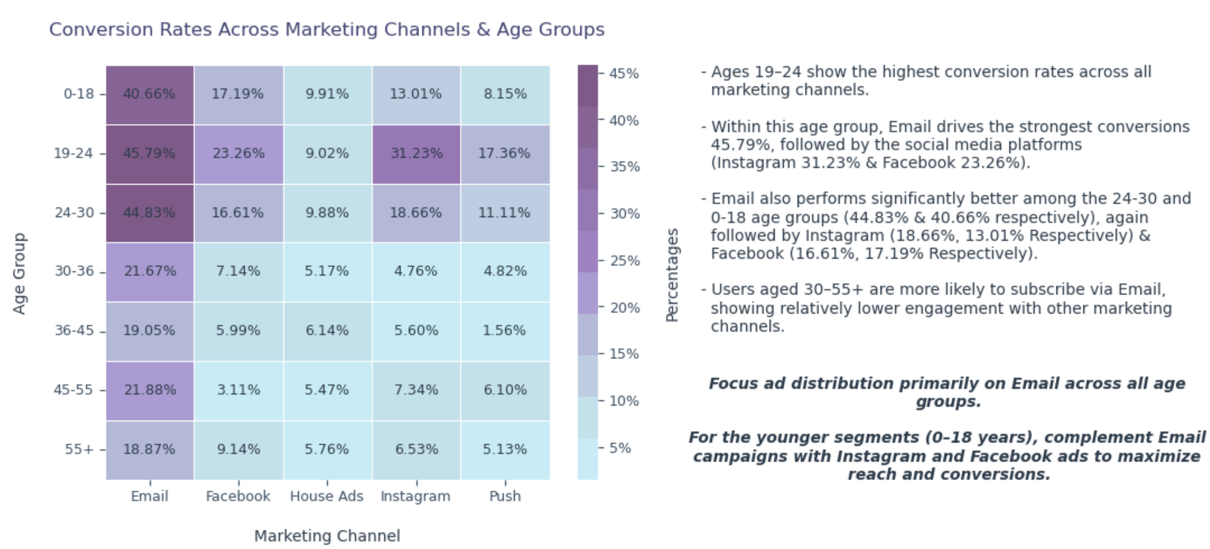
In [565…

```
# Heatmap Function:
def heatmap_chart(df):
    col_list=df.columns.to_list()
    result_pivot=df.pivot_table(values=col_list[-1], index=col_list[1],columns=col_

    palette_10 = sns.color_palette(['#C9EFF5', '#C5E2ED', '#BECFE3', '#B4BADA', '#A

    # Creating the Chart:
    ax=sns.heatmap(result_pivot, cmap=palette_10, annot=True, fmt=".2%", annot_kws=
                   linewidths=.4)

    # Customizing the Chart:
    plt.title('', fontsize=12, color='#454775')

    cbar = ax.collections[0].colorbar
    cbar.ax.yaxis.set_major_formatter(mticker.FuncFormatter(lambda x, pos: f'{x*100
    cbar.set_label('\nPercentages', fontsize=10, color='#313E4c')
    cbar.ax.tick_params(labelsize=9, colors='#415366')

    ax.set_xlabel(f'\n{col_list[0]}',fontsize=10, color='#313E4c')
    ax.set_ylabel(f'{col_list[1]}\n',fontsize=10, color='#313E4c')
    ax.tick_params(axis='both',labelsize=9, colors='#415366',labelrotation=0)
```

In [566…

```
# Conversion Rates Across Marketing Channels & Age Groups
conversion_ch_age=con_ret(marketing,converted_users,['marketing_channel','age_group

conversion_ch_age['Age Group']=[x.replace(' years','') for x in conversion_ch_age['

# Visualization
heatmap_chart(conversion_ch_age)
```

```
# Additional Customization:
plt.title('\nConversion Rates Across Marketing Channels & Age Groups\n')

# Findings:
text_ch_a_con=f'''
- Ages 19–24 show the highest conversion rates across all
  marketing channels.\n
- Within this age group, Email drives the strongest conversions
  {conversion_ch_age.sort_index().iloc[1,4]:.2%}, followed by the social media plat
  (Instagram {conversion_ch_age.sort_index().iloc[22,4]:.2%} & Facebook {conversion
- Email also performs significantly better among the 24-30 and
  0-18 age groups ({conversion_ch_age.sort_index().iloc[2,4]:.2%} & {conversion_ch_
  followed by Instagram ({conversion_ch_age.sort_index().iloc[23,4]:.2%}, {conversi
  Facebook ({conversion_ch_age.sort_index().iloc[9,4]:.2%}, {conversion_ch_age.sort
- Users aged 30–55+ are more likely to subscribe via Email,
  showing relatively lower engagement with other marketing
  channels.'''

text2_ch_a_con = '''
Focus ad distribution primarily on Email across all age
groups.\n
For the younger segments (0–18 years), complement Email
campaigns with Instagram and Facebook ads to maximize
reach and conversions.'''

plt.text(6.7,4.5,text_ch_a_con, color='#313E4C', ha='left')
plt.text(9.5,7,text2_ch_a_con, ha='center',fontstyle='italic',weight='semibold', fo

plt.show()
```



$Q4$: ***How do ad type and user age interact to influence conversion and retention rates?***

In [567…

```
# Conversion Rates for the combinations of Ad Classification & Age groups
conversion_var_age=con_ret(marketing,converted_users,['variant','age_group'],'user_
```

```python
conversion_var_age['Age Group']=[x.replace(' years','') for x in conversion_var_age

conversion_var_age['Variant']=conversion_var_age['Variant'].apply(lambda x: x.title

# 1- Visualiztion - Conversion Rates for the combinations of Ad Classification & Ag
heatmap_chart(conversion_var_age)
# Additional Customization:
plt.title('\nConversion Rates Across Ad Classification & Age Groups\n')

# Findings
text_var_age_c=f'''
- Users under 30 show a much higher tendency to subscribe,
  especially when exposed to personalized ads, with an average
  conversion rate of {np.mean(conversion_var_age.sort_values(['Variant','Age Group'
  controlled ads.\n
- For users aged 30 and above, conversion rates drop notably.
  In this segment, control ads actually perform about twice
  as well as personalized ads.'''

text2_var_age_c= '''
Personalized ads are highly effective among younger
audiences (under 30).\n
while simpler, non-personalized messages may resonate
better with older users (30+).'''

plt.text(2.7,3,text_var_age_c, color='#313E4C', ha='left')
plt.text(3.9,6,text2_var_age_c, ha='center',fontstyle='italic',weight='semibold', f

plt.show()

# Retention Rates for the combinations of Ad Classification & Age groups
retention_var_age=con_ret(converted_users,retained_users,['variant','age_group'],'u

retention_var_age['Age Group']=[x.replace(' years','') for x in retention_var_age['

retention_var_age['Variant']=retention_var_age['Variant'].apply(lambda x: x.title()

# 1- Visualiztion - Conversion Rates for the combinations of Ad Classification & Ag
heatmap_chart(retention_var_age)
# Customizing the Chart:
plt.title('\nRetention Rates Across Ad Classification & Age Groups\n')

# Findings
text_var_age_r=f'''
- Ages 30-36 shows the highest retention rate via controlled
  ads {retention_var_age.sort_values(['Variant','Age Group']).iloc[3,4]:.2%} & the
  ads {retention_var_age.sort_values(['Variant','Age Group']).iloc[10,4]:.2%}.\n
- Retention rates are generally higher for the control variant
  across all age groups.\n
- A noticable gap between controlled and personalized ads
  retention rates for ages of 30 & above. \n
- Younger users (below 30) display moderate retention for
  both ad types, with differences between variants being
  relatively small.'''
```

```
text2_var_age_r= '''
While personalization increases conversion among
younger users, it appears to reduce long-term
retention, especially for older users. \n
A mixed strategy—personalized content for acquisition
and standard communication for retention—may achieve
better overall performance.'''

plt.text(2.7,3.5,text_var_age_r, color='#313E4C', ha='left')
plt.text(3.8,6.5,text2_var_age_r, ha='center',fontstyle='italic',weight='semibold',

plt.show()
```