# Question3:

Some exploration of data I would do before finding out the most performant popup version are

* Analyse null values, get counts and filter out null records in pop up version
* Analyse outliers in views and registrations. Below are some examples which I looked at

On 13/12/2019 null popup version and Character Profile Checklist 4 popup name we have 33880 views and 5397 registrations which is an outlier. We need to find out what happened on that day.

On 16/10/2019 null popup version and Worldbuilding Template popup name we have 17248 views and 1897 registrations which is another outlier

482 out of 2063 records views outliers, 283 out of 2083 records registrations outliers observed and majority of these outlier occurances are in 2020 experiment start dates compared to previous years, that may be due to more marketing during covid? or people have more time to write books?

* 1543 out of 2063 records have zero registrations although sometimes views have been around 15000. Reason for why the views have not been translated to registrations have to investigated.
* Looking at correlation of views and registrations we could see views are not much correlated to registrations.

|  | **views** | **registrations** |
| --- | --- | --- |
| views | 1.00000 | 0.47493 |
| registrations | 0.47493 | 1.00000 |

* descriptive analysis can be performed, like pop-up distribution per category, date etc. Also, aggregations like total number of views or registration per category or date

**Code snippets:**

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

df=pd.read\_csv('dataset.tsv', sep='\t')

# Split the column into three separate one:

df = pd.DataFrame(df['popup\_version|start\_date|popup\_category'].to\_list(), columns=['popup\_version','start\_date','popup\_category'])

#Data Cleaning: Clean the data and handle missing values or outliers if any.

df.dropna(inplace=True)

#Filter A/B Tests: Filter the data to only include rows with populated popup\_version values (i.e., only rows that are part of an A/B test)

df['popup\_version'].replace({'': np.nan}, inplace=True)

df1 = df[pd.notna(df['popup\_version'])]

df1.groupby('popup\_version').count()

#Create a new column called conversion that calculates the conversion rate as the ratio of registrations to views

df1['conversion\_rate'] = df1['registrations'] / df1['views']

#Group by A/B Test and Version: Group the data by the start date of the A/B test and the popup\_version column.

# This will give us the conversion rate for each A/B test and popup version

df\_grouped = df1.groupby(['start\_date','popup\_name', 'popup\_version'], as\_index=False).agg({'conversion\_rate': 'mean'})

# Use the 'apply' method to join the values in the 'popup\_version' column

df\_grouped = df\_grouped.groupby(['start\_date','popup\_name', 'conversion\_rate']).apply(lambda x: ', '.join(x['popup\_version']))

df\_grouped = df\_grouped.reset\_index(name='popup\_version')

print(df\_grouped.head(10))

#Find the Most Performant Popup Version: For each A/B test, find the popup version with the highest conversion rate.

df\_grouped = df\_grouped.sort\_values(by=['start\_date', 'conversion\_rate'], ascending=[True, False])

print(df\_grouped.head(10))

df\_grouped = df\_grouped.groupby('start\_date').first().reset\_index()

df\_grouped.head(10)

#Show the Results: The popup\_version column now contains the most performant popup version for each A/B test.

# You can now show the corresponding conversion rate for each A/B test.

print(df\_grouped[['start\_date','popup\_name','popup\_version', 'conversion\_rate']])

# The resulting dataframe df\_grouped will contain the start date of each A/B test, the most performant popup version,and the corresponding conversion rate. This information can be used to make decisions about which popup version to use in future tests to improve conversion rates and achieve Bookly's goal of converting page views into registration

# Question4:

Code snippets:

#After reading the data as in previous question, create a grouped data

df\_grouped = df.groupby(['start\_date'], as\_index=False).agg({'registrations': 'sum', 'views': 'sum'})

df\_grouped['conversion\_rate'] = df\_grouped['registrations'] / df\_grouped['views']

df\_grouped.to\_csv("conversion\_rate\_per\_start\_date.csv",header=True, index = False)

# this csv can be found in the repository

# Specify the date format

date\_format = "%d/%m/%Y"

# Convert the start\_date column to datetime object with the specified format

df\_grouped["start\_date"] = pd.to\_datetime(df\_grouped["start\_date"], format=date\_format)

# set start\_date as the index

df\_grouped.set\_index("start\_date", inplace=True)

df\_grouped = df\_grouped.sort\_values(by='start\_date')

print(df\_grouped.head(20))

# Plot the line graph

df.plot(x="start\_date", y="conversion\_rate", kind="line")

# rotate x-axis labels

plt.xticks(rotation=90)

# Set X axis as date axis and format it

plt.gcf().autofmt\_xdate()

# Show the plot

plt.show()

# chart is shared in repository as well