TikTok Project

Course 2 - Get Started with Python

Welcome to the TikTok Project!

You have just started as a data professional at TikTok.

The team is still in the early stages of the project. You have received notice that TikTok's leadership team has approved the project proposal. To gain clear insights to prepare for a claims classification model, TikTok's provided data must be examined to begin the process of exploratory data analysis (EDA).

A notebook was structured and prepared to help you in this project. Please complete the following questions.

Course 2 End-of-course project: Inspect and analyze data

In this activity, you will examine data provided and prepare it for analysis.

The purpose of this project is to investigate and understand the data provided. This activity will:

1. Acquaint you with the data
2. Compile summary information about the data
3. Begin the process of EDA and reveal insights contained in the data
4. Prepare you for more in-depth EDA, hypothesis testing, and statistical analysis

The goal is to construct a dataframe in Python, perform a cursory inspection of the provided dataset, and inform TikTok data team members of your findings.

This activity has three parts:
Part 1: Understand the situation

- How can you best prepare to understand and organize the provided TikTok information?

Part 2: Understand the data

- Create a pandas dataframe for data learning and future exploratory data analysis (EDA) and statistical activities
- Compile summary information about the data to inform next steps

Part 3: Understand the variables

- Use insights from your examination of the summary data to guide deeper investigation into variables

To complete the activity, follow the instructions and answer the questions below. Then, you will us your responses to these questions and the questions included in the Course 2 PACE Strategy Document to create an executive summary.

Be sure to complete this activity before moving on to Course 3. You can assess your work by comparing the results to a completed exemplar after completing the end-of-course project.

Identify data types and compile summary information

Task 1. Understand the situation

- How can you best prepare to understand and organize the provided information?

Begin by exploring your dataset and consider reviewing the Data Dictionary.

Response: Prepare by reading in the data, viewing the data dictionary, and exploring the dataset to identify key variables for the stakeholder.

Task 2a. Imports and data loading

Start by importing the packages that you will need to load and explore the dataset. Make sure to use the following import statements:

- `import pandas as pd`
- `import numpy as np`

In [1]:
```
# Import packages
import pandas as pd
import numpy as np
```

Then, load the dataset into a dataframe. Creating a dataframe will help you conduct data manipulation, exploratory data analysis (EDA), and statistical activities.

Note: As shown in this cell, the dataset has been automatically loaded in for you. You do not need to download the .csv file, or provide more code, in order to access the dataset and proceed with this lab. Please continue with this activity by completing the following instructions.

In [2]:
```
data = pd.read_csv("tiktok_dataset.csv")
```

Task 2b. Understand the data - Inspect the data

View and inspect summary information about the dataframe by **coding the following:**

1. `data.head(10)`
2. `data.info()`
3. `data.describe()`

Consider the following questions:

Question 1: When reviewing the first few rows of the dataframe, what do you observe about the data? What does each row represent?

Question 2: When reviewing the `data.info()` output, what do you notice about the different variables? Are there any null values? Are all of the variables numeric? Does anything else stand out?

Question 3: When reviewing the `data.describe()` output, what do you notice about the distributions of each variable? Are there any questionable values? Does it seem that there are outlier values?

In [3]:
```
# Display and examine the first 10 rows of the dataframe

data.head(10)
```

Out[3]:

	#	claim_status	video_id	video_duration_sec	video_transcription_text	verifie
0	1	claim	7017666017	59	someone shared with me that drone deliveries a...	n
1	2	claim	4014381136	32	someone shared with me that there are more mic...	n
2	3	claim	9859838091	31	someone shared with me that american industria...	n
3	4	claim	1866847991	25	someone shared with me that the metro of st. p...	n
4	5	claim	7105231098	19	someone shared with me that the number of busi...	n
5	6	claim	8972200955	35	someone shared with me that gross domestic pro...	n
6	7	claim	4958886992	16	someone shared with me that elvis presley has ...	n
7	8	claim	2270982263	41	someone shared with me that the best selling s...	n
8	9	claim	5235769692	50	someone shared with me that about half of the ...	n
9	10	claim	4660861094	45	someone shared with me that it would take a 50...	

In [4]:
```
# Get summary info

data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 19382 entries, 0 to 19381
Data columns (total 12 columns):
 #   Column                    Non-Null Count  Dtype  
---  ------                    --------------  -----  
 0   #                         19382 non-null  int64  
 1   claim_status              19084 non-null  object 
 2   video_id                  19382 non-null  int64  
 3   video_duration_sec        19382 non-null  int64  
 4   video_transcription_text  19084 non-null  object 
 5   verified_status           19382 non-null  object 
 6   author_ban_status         19382 non-null  object 
 7   video_view_count          19084 non-null  float64
 8   video_like_count          19084 non-null  float64
 9   video_share_count         19084 non-null  float64
 10  video_download_count      19084 non-null  float64
 11  video_comment_count       19084 non-null  float64
dtypes: float64(5), int64(3), object(4)
memory usage: 1.8+ MB
```

In [5]:
```
# Get summary statistics

data.describe()
```

Out[5]:

	#	video_id	video_duration_sec	video_view_count	video_like
count	19382.000000	1.938200e+04	19382.000000	19084.000000	19084.(
mean	9691.500000	5.627454e+09	32.421732	254708.558688	84304.(
std	5595.245794	2.536440e+09	16.229967	322893.280814	133420.
min	1.000000	1.234959e+09	5.000000	20.000000	0.(
25%	4846.250000	3.430417e+09	18.000000	4942.500000	810.1
50%	9691.500000	5.618664e+09	32.000000	9954.500000	3403.5
75%	14536.750000	7.843960e+09	47.000000	504327.000000	125020.(
max	19382.000000	9.999873e+09	60.000000	999817.000000	657830.(

Response:

Question 1: The dataframe contains a collection of categorical, text, and numerical data. Each row represents a distinct TikTok video that presents either a claim or an opinion and the accompanying metadata about that video.

Question 2: The dataframe contains five float64s, three int64s, and four objects. There are 19,382 observations, but some of the variables are missing values, including claim status, the video transcripton, and all of the count variables.

Question 3: Many of the count variables seem to have outliers at the high end of the distribution. They have very large standard deviations and maximum values that are very high compared to their quartile values.

Task 2c. Understand the data - Investigate the variables

In this phase, you will begin to investigate the variables more closely to better understand them.

You know from the project proposal that the ultimate objective is to use machine learning to classify videos as either claims or opinions. A good first step towards understanding the data might therefore be examining the `claim_status` variable. Begin by determining how many videos there are for each different claim status.

In [6]:
```
# What are the different values for claim status and how many of each are ir

data['claim_status'].value_counts()
```

Out[6]:
```
claim      9608
opinion    9476
Name: claim_status, dtype: int64
```

Response: The counts of each claim status are quite balanced.

Next, examine the engagement trends associated with each different claim status.

Start by using Boolean masking to filter the data according to claim status, then calculate the mean and median view counts for each claim status.

In [7]:
```
# What is the average view count of videos with "claim" status?

claims = data[data['claim_status'] == 'claim']
print('Mean view count claims:', claims['video_view_count'].mean())
print('Median view count claims:', claims['video_view_count'].median())
```

```
Mean view count claims: 501029.4527477102
Median view count claims: 501555.0
```

In [8]:
```
# What is the average view count of videos with "opinion" status?

opinions = data[data['claim_status'] == 'opinion']
print('Mean view count opinions:', opinions['video_view_count'].mean())
print('Median view count opinions:', opinions['video_view_count'].median())
```

```
Mean view count opinions: 4956.43224989447
Median view count opinions: 4953.0
```

Response: The mean and the median within each claim category are close to one another, but there is a vast discrepancy between view counts for videos labeled as claims and videos labeled as opinions.

Now, examine trends associated with the ban status of the author.

Use `groupby()` to calculate how many videos there are for each combination of categories of claim status and author ban status.

In [9]:
```
# Get counts for each group combination of claim status and author ban statu

data.groupby(['claim_status', 'author_ban_status']).count()[['#']]
```

Out[9]:

claim_status	author_ban_status	#
claim	**active**	6566
	banned	1439
	under review	1603
opinion	**active**	8817
	banned	196
	under review	463

Response: There are many more claim videos with banned authors than there are opinion videos with banned authors. This could mean a number of things, including the possibilities that:

- Claim videos are more strictly policed than opinion videos
- Authors must comply with a stricter set of rules if they post a claim than if they post an opinion

Also, it should be noted that there's no way of knowing if claim videos are inherently more likely than opinion videos to result in author bans, or if authors who post claim videos are more likely to post videos that violate terms of service.

Finally, while you can use this data to draw conclusions about banned/active authors, you cannot draw conclusions about banned videos. There's no way of determining whether a particular video *caused* the ban, and banned authors could have posted videos that complied with the terms of service.

Continue investigating engagement levels, now focusing on `author_ban_status`.

Calculate the median video share count of each author ban status.

In [10]:
```
data.groupby(['author_ban_status']).agg(
    {'video_view_count': ['mean', 'median'],
     'video_like_count': ['mean', 'median'],
     'video_share_count': ['mean', 'median']})
```

Out[10]:

	video_view_count		video_like_count		video_sha
	mean	median	mean	median	mean
author_ban_status					
active	215927.039524	8616.0	71036.533836	2222.0	14111.466164
banned	445845.439144	448201.0	153017.236697	105573.0	29998.942508
under review	392204.836399	365245.5	128718.050339	71204.5	25774.696999

In [11]:
```
# What's the median video share count of each author ban status?

data.groupby(['author_ban_status']).median(numeric_only=True)[
    ['video_share_count']]
```

Out[11]:

	video_share_count
author_ban_status	
active	437.0
banned	14468.0
under review	9444.0

Response: Banned authors have a median share count that's 33 times the median share count of active authors! Explore this in more depth.

Use `groupby()` to group the data by `author_ban_status`, then use `agg()` to get the count, mean, and median of each of the following columns:

- `video_view_count`
- `video_like_count`
- `video_share_count`

Remember, the argument for the `agg()` function is a dictionary whose keys are columns. The values for each column are a list of the calculations you want to perform.

In [12]:
```
data.groupby(['author_ban_status']).agg(
    {'video_view_count': ['count', 'mean', 'median'],
     'video_like_count': ['count', 'mean', 'median'],
     'video_share_count': ['count', 'mean', 'median']
     })
```

Out[12]:

	video_view_count			video_like_count		
	count	mean	median	count	mean	median
author_ban_status						
active	15383	215927.039524	8616.0	15383	71036.533836	2222.0
banned	1635	445845.439144	448201.0	1635	153017.236697	105573.0
under review	2066	392204.836399	365245.5	2066	128718.050339	71204.5

Response: A few observations stand out:

- Banned authors and those under review get far more views, likes, and shares than active authors.
- In most groups, the mean is much greater than the median, which indicates that there are some videos with very high engagement counts.

Now, create three new columns to help better understand engagement rates:

- `likes_per_view`: represents the number of likes divided by the number of views for each video
- `comments_per_view`: represents the number of comments divided by the number of views for each video
- `shares_per_view`: represents the number of shares divided by the number of views for each video

In [13]:
```
# Create a likes_per_view column
data['likes_per_view'] = data['video_like_count'] / data['video_view_count']

# Create a comments_per_view column
data['comments_per_view'] = data['video_comment_count'] / data['video_view_c

# Create a shares_per_view column
data['shares_per_view'] = data['video_share_count'] / data['video_view_count
```

Use `groupby()` to compile the information in each of the three newly created columns for each combination of categories of claim status and author ban status, then use `agg()` to calculate the count, the mean, and the median of each group.

In [14]:
```
data.groupby(['claim_status', 'author_ban_status']).agg(
    {'likes_per_view': ['count', 'mean', 'median'],
     'comments_per_view': ['count', 'mean', 'median'],
     'shares_per_view': ['count', 'mean', 'median']})
```

Out[14]:

		likes_per_view			comments_per_vi	
		count	mean	median	count	mean	med
claim_status	author_ban_status						
claim	active	6566	0.329542	0.326538	6566	0.001393	0.0007
	banned	1439	0.345071	0.358909	1439	0.001377	0.0007
	under review	1603	0.327997	0.320867	1603	0.001367	0.0007
opinion	active	8817	0.219744	0.218330	8817	0.000517	0.0002
	banned	196	0.206868	0.198483	196	0.000434	0.0001
	under review	463	0.226394	0.228051	463	0.000536	0.0002

Response: We know that videos by banned authors and those under review tend to get far more views, likes, and shares than videos by non-banned authors. However, *when a video does get viewed*, its engagement rate is less related to author ban status and more related to its claim status.

Also, we know that claim videos have a higher view rate than opinion videos, but this tells us that claim videos also have a higher rate of likes on average, so they are more favorably received as well. Furthermore, they receive more engagement via comments and shares than opinion videos.

Note that for claim videos, banned authors have slightly higher likes/view and shares/view rates than active authors or those under review. However, for opinion videos, active authors and those under review both get higher engagement rates than banned authors in all categories.

Given your efforts, what can you summarize for Rosie Mae Bradshaw and the TikTok data team?

Note for Learners: Your answer should address TikTok's request for a summary that covers the following points:

- What percentage of the data is comprised of claims and what percentage is comprised of opinions?
- What factors correlate with a video's claim status?
- What factors correlate with a video's engagement level?

Response:

- Of the 19,382 samples in this dataset, just under 50% are claims—9,608 of them.

- Engagement level is strongly correlated with claim status. This should be a focus of further inquiry.
- Videos with banned authors have significantly higher engagement than videos with active authors. Videos with authors under review fall between these two categories in terms of engagement levels.