Initial Thoughts:

I entered this project wanting to learn the relationship between the average likes, shares, comments, and views and a Tiktok creator's follower count. These measures of engagement can be important for brands to determine if a creator is suitable for a brand deal just based on engagement metrics as well. Learning what follower count gets the most engagement per video, I can determine which group of Tiktokers I should look at when determining which I should reach out to about a sponsorship. I want my product to reach as many people as possible, but I want these people to also actually engage with my advertisement. This is just a hypothetical, but I wanted to show how this data could be actually applicable. For the sake of this project, I just want to see if there is a trend in any of this data when graphed on a 2D plane.

Writing the Code:

Data Processing:

The dataset I got from Kaggle had some issues. For starters, all of the values under the metrics I was interested in looking at were all strings. 'K' denoted a thousand and 'M' denoted a million. This would make my data impossible to work with unless I found a way to clean it up. I made a method that would read the CSV file. First, I created the public struct for my CSVFileProcessor. First, I made sure that all of my headers would remain intact. I let Regex clean up my numbers first. Regex is an important tool for parsing text and cleaning numbers. I then set up my method so that it would take an input of my CSV file, then return a CSV file with clean values. I cleaned my values by multiplying any value that had a 'K' or 'k' by a thousand. I did the same for any with a 'M' or 'm', but by a million. This ensured that I had good data to work with.

K_Means:

I thought that k_means would be the best way to group influencers and their metrics. I had some hiccups when working on my k_means code however. I first established a trait where I supported each of my methods. I made one method to calculate my kmeans, one to intialize my centroids, one to update centroids, one to update labels, and one to calculate the euclidean distance between points. These functions all work together to calculate the k_means of the metrics. My code will iterate through each influencer to determine what center they are closest to. I make another public function to actually plot my data.

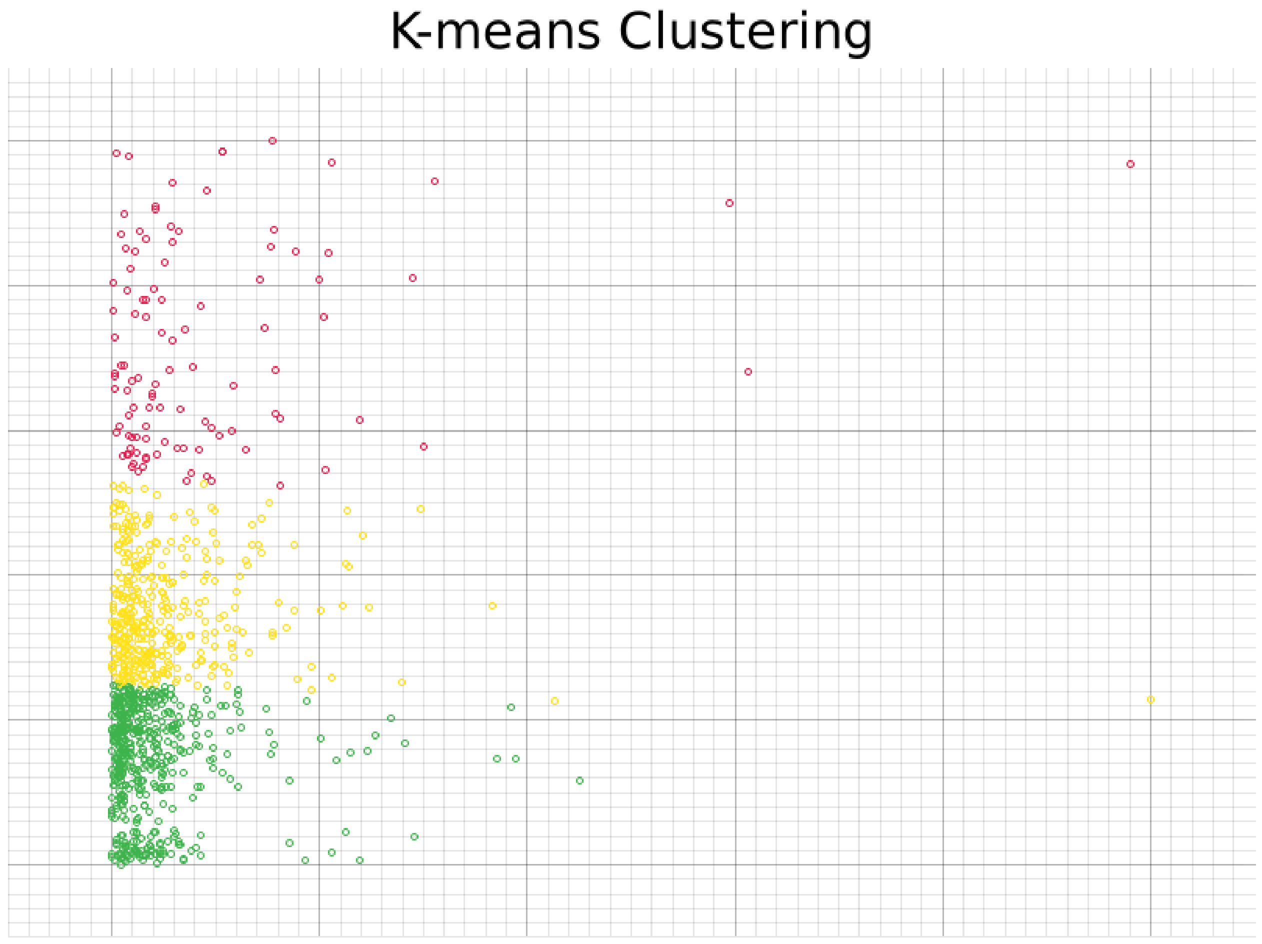
Normalizing Data:

Something I needed to do since I was working with such large sized data was to normalize it. I also needed to change all my values into floating points so that I could find the k-means. I make a function that when called will return a vector of floating point vectors when a dataframe and its column name is the input.

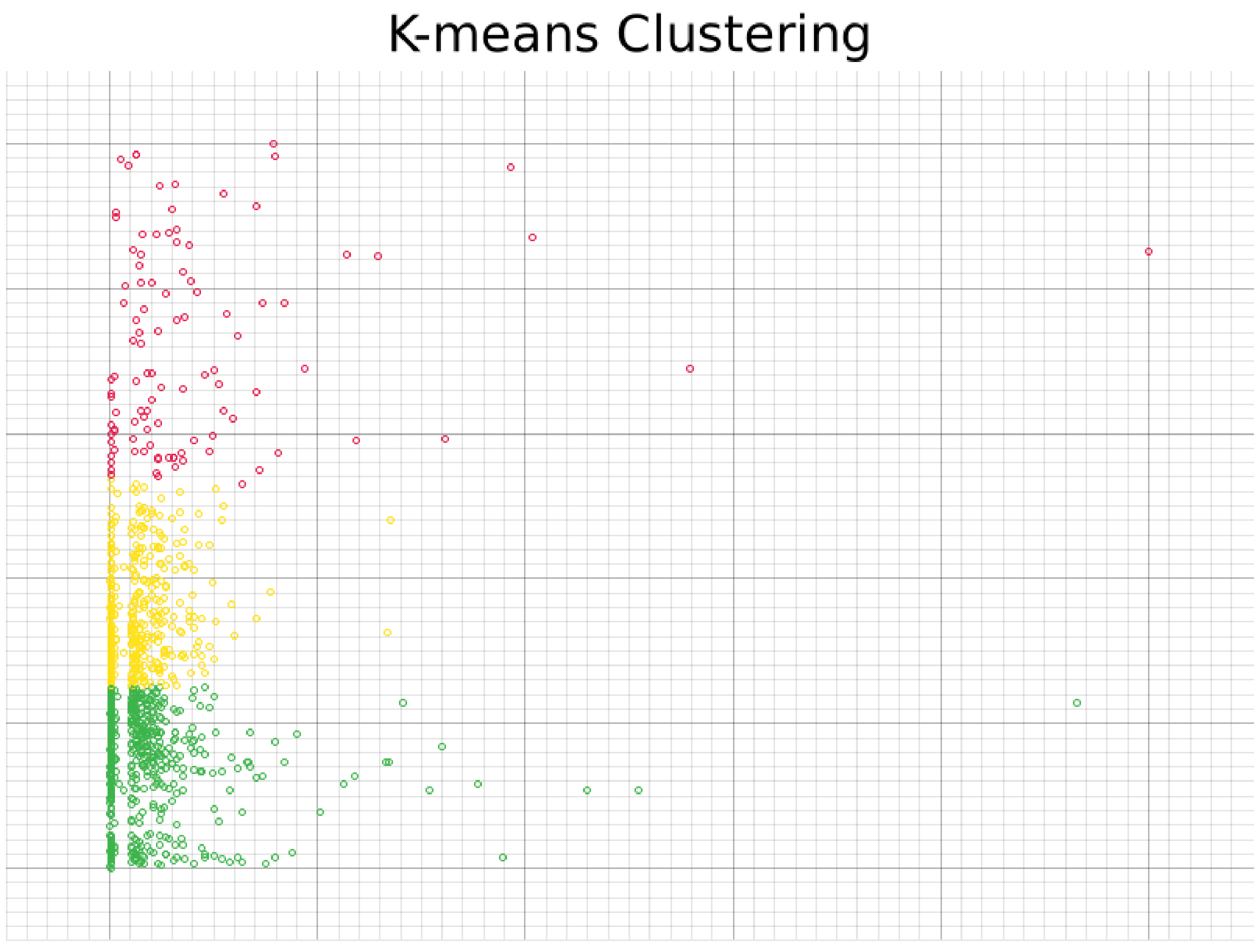
Main Function:

My main function processes each of the functions and methods I’ve created. I first read and clean my csv file in the main function using the processing_Data module I made. Next, I read the cleaned data with polar Dataframe. This way my data is represented in an easy to read visual. Next, I used my function that normalized my columns. I stored each of the vectors to their respective column type (vector containing average likes is named likes, etc.) Next, I create a truly 2D array by iterating over both vectors I selected until they were joint. Now that they were joint, I could properly run my kmeans algorithm over it. The kmeans produced my labels, so the next thing I had to do was produce my graph. Here are my graphs of:

Followers vs Likes:

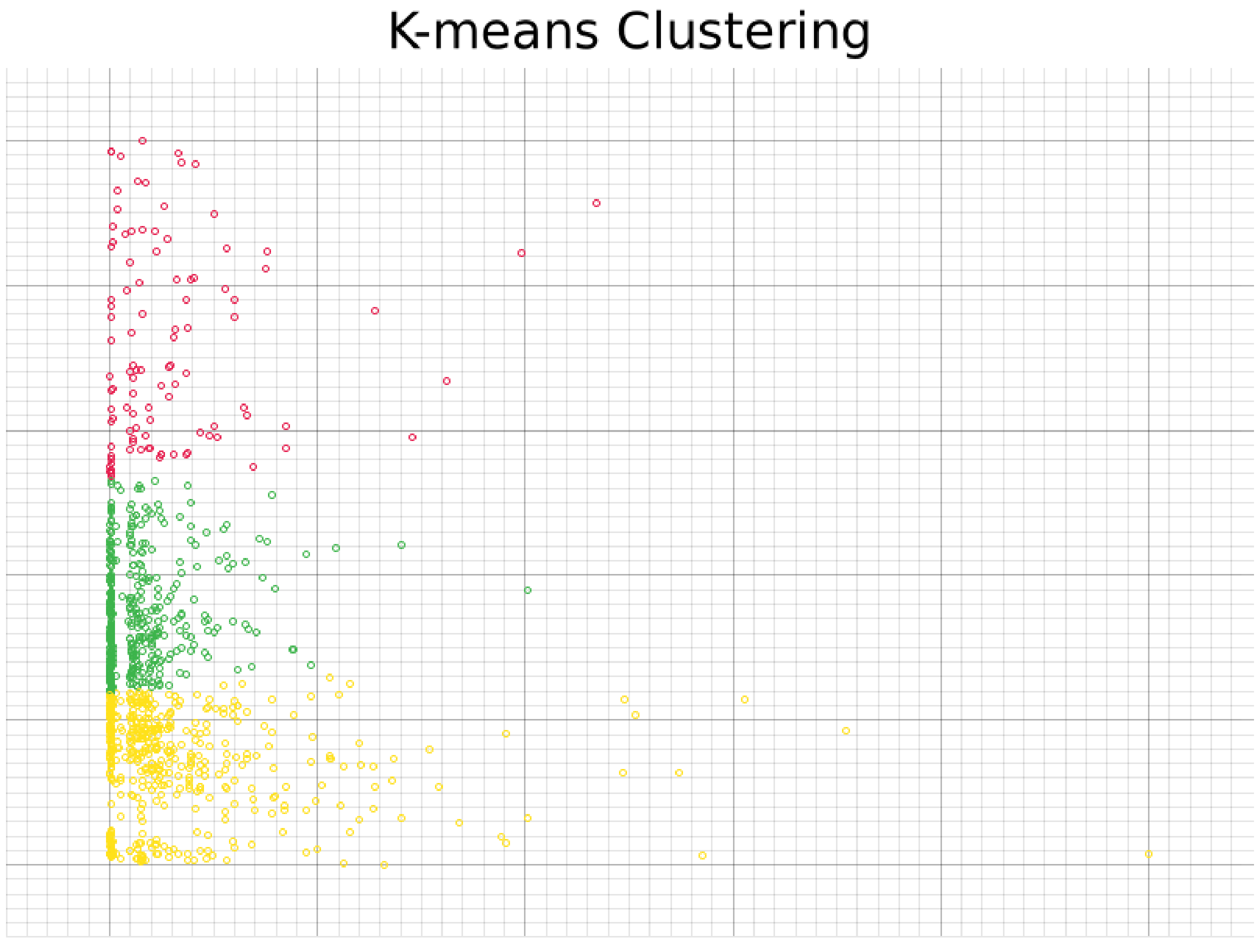


Comments vs. Likes:



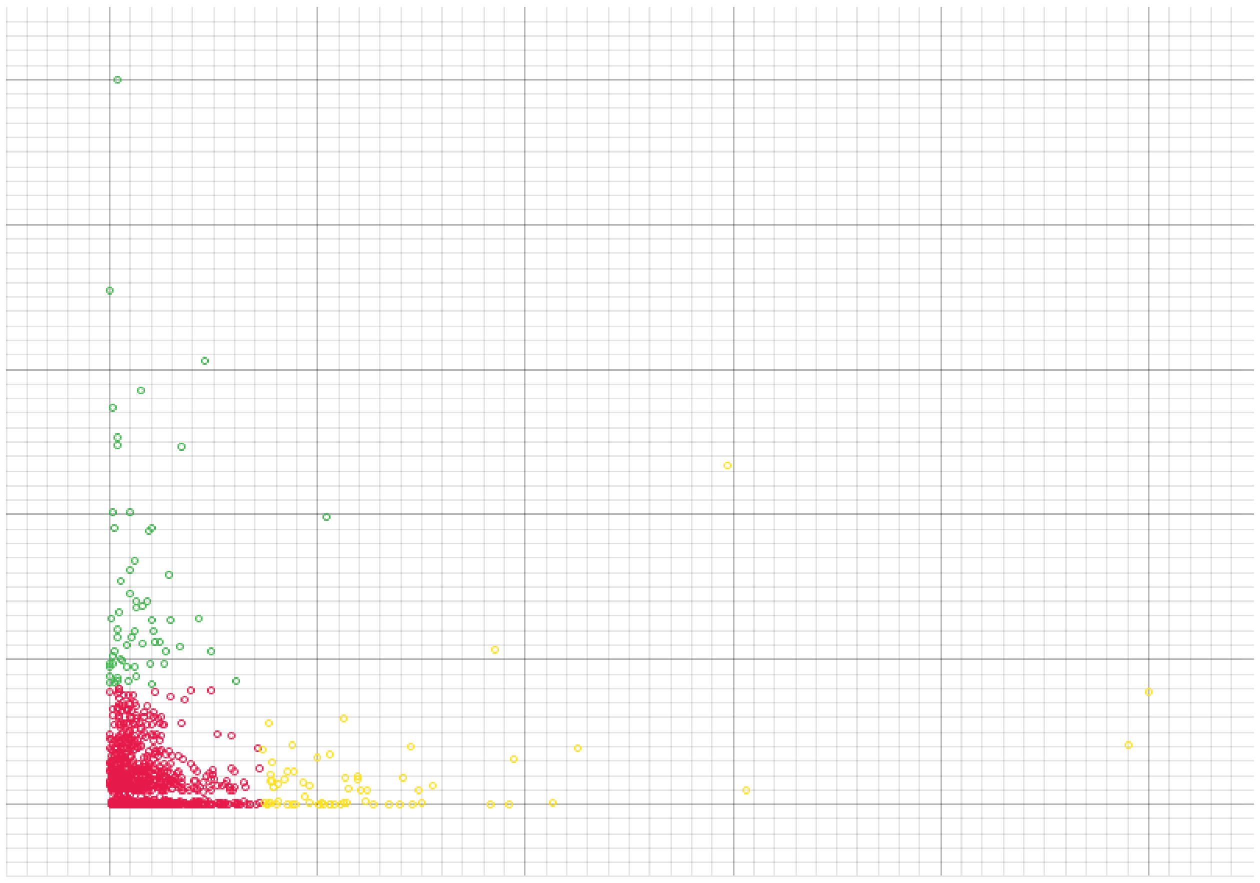
I want to quickly talk about this graph, because I find it the most fascinating. There are quite a few posts with high amounts of likes, but lower amounts of comments. These are the red values. The green values have low amounts of likes and lower amounts of comments. Posts with absurdly high likes tend to have fewer comments proportionally according to this data. There is an outlier that is really interesting: one Tiktok influencer has a higher ratio of comments to likes. There is another Tiktoker who receives more comments than likes on average. In my experience, when a tiktok has more comments than dislikes, the video is usually very controversial. Perhaps this is a controversial Tiktok influencer that is receiving more comments than likes. However, this is just a personal anecdote tied to something that I thought was interesting. When I work a little further on this project over break, I want to see who it is. Based on these trends however, I see that there seems that most users have more likes than comments on their videos, most Tiktok influencers tend to have the same amount of comments.

Shares vs Likes:

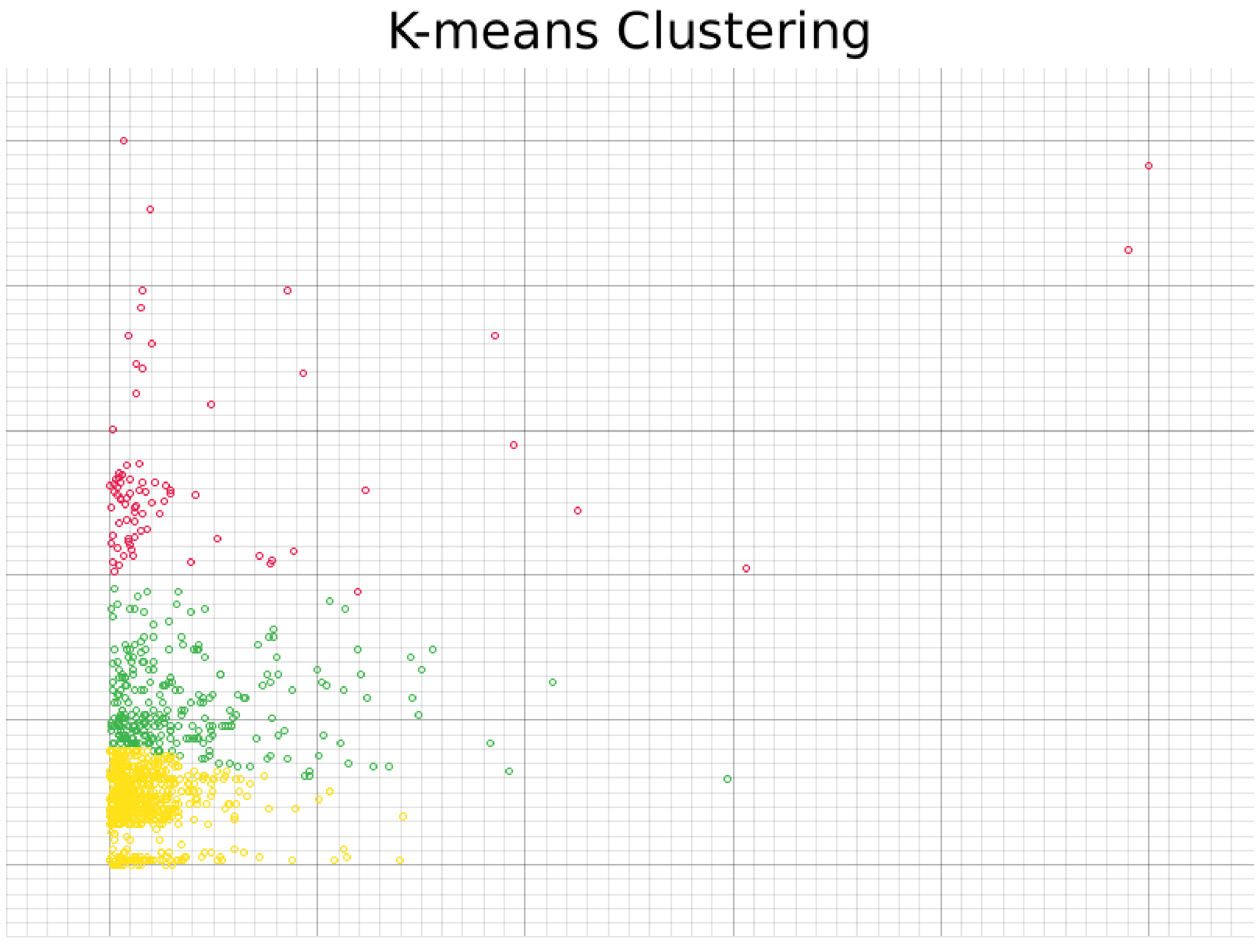


Followers vs. Shares:

K-means Clustering



Followers vs. Views:



This graph is also fascinating. It shows that some groups with smaller follower amounts do have a greater concentration of views, while some Tiktok influencers with a large amount of followers get very few views. There are a couple outliers that say the opposite however. There is an interesting pattern here that shows that even if you have a smaller or average amount of followers, you can still have a greater amount of views on average than creators with an incredible amount of followers.

Running This Code Again:

If someone wanted to run this code against another data set of similar parameters, they would first call the clean_csv method to clean up their data and return it in another file. Next, they would use the polar CsvReader::from_path to display the now cleaned dataframe so that the user can gauge a visual. Next, the user would clone the data frame so that it was mutable. Next, the user would declare each column they would want to change from integers to floating points by calling extract_floats_from_columns function. They would name each of their vectors containing the floating points by their column name, or something similar. Next, the user would make a Vec<Vec<f64>> by combining two of the different vectors we just made in one vector. The user would then declare their labels depending on their value of k. Lastly, the user would call their plot_data for the new Vec<Vec<f64>> they made. This cycle can be repeated to compare any two different types of engagement averages or to compare follower count to an engagement average.

For the Future:

There are some things that I did experiment with, but could not figure out on my own. I would have really liked for this project to be done completely with polars. I am sure it could have, especially k_means, but it was beyond the scope of this class. I want to come back to this project because I thought that the prompt I came up with was interesting. I think that looking at social media engagement is important for trend tracking. Additionally, I previously had a function that would analyze my data. It would provide the median, mean, and count for each of the different columns. That function worked initially, but after some of the changes I made, I could not rework it to fit my code. Though it provides some insight, it does not provide enough to really matter. If you are interested, the output is added below. I really do want to work on this a little over break, so I will keep the code and see how I can add onto it. This sort of stuff is fascinating to me. It will keep me very busy over break. Have a great break and happy holidays!

Use of Generative AI and Secondary Sources:

There were many times I used ChatGPT to help fix my code. There were a few things I was using that we had not delved into during class time, but still made my project worthwhile. I really wanted to generate a visual representation for this project, so I used k-means to show it. However, I would also like to add that a lot of the generative code I initially got from ChatGPT changed drastically over time from my own edits. I start from just plain CSV reading, then go to using polar DataFrame, then move from using a horrible k-means library to creating my own way of calculating k-means. I used AI to add to my learning, edit code, and to help read error messages. I have a hard time understanding the few words an error message gives me, so I paste the error message into ChatGPT so that I can actually understand what is going on with the code. There are some technical terms that I've learned a bit more about from doing this. I am able to decode some error messages on my own now, which has made this entirely more helpful.

https://chatgpt.com/share/675f6eaf-5dfc-8011-8c2c-f47daf8c3ec1

https://chatgpt.com/share/675f6f0f-801c-8011-9ad9-82c0d989c898

https://chatgpt.com/share/675f6f1d-35e0-8011-856a-cb0ce4d224ec

In addition, I used a couple secondary sources to help me understand more about Rust syntax, different libraries, and to learn a couple of ways I could write my code.

https://users.rust-lang.org/t/operator-has-incompatible-types/81564

https://docs.pola.rs/user-guide/expressions/casting/#basic-example

https://docs.pola.rs/

Extra data, just for interest:

Column: S.no, Mean: 500.50, Median: 500.50, Count: 1000

Column: Tiktoker name, Mean: 0.00, Median: 0.00, Count: 1000

Column: Tiktok name, Mean: 0.67, Median: 0.00, Count: 1000

Column: Subscribers, Mean: 7083611.30, Median: 3500000.00, Count: 1000

Column: Views avg., Mean: 2854531.40, Median: 2200000.00, Count: 1000

Column: Likes avg., Mean: 352434.50, Median: 274700.00, Count: 1000

Column: Comments avg., Mean: 2540.13, Median: 1600.00, Count: 1000

Column: Shares avg., Mean: 3263.48, Median: 1800.00, Count: 1000

DataFrame Summary: (NOVEMBER)
Column: row-cell, Mean: 500.00, Median: 500.00, Count: 1001
Column: Tiktoker name, Mean: 0.00, Median: 0.00, Count: 1001
Column: Tiktok name, Mean: 0.00, Median: 0.00, Count: 1001
Column:
Followers, Mean: 8200511.29, Median: 4700000.00, Count: 1001
Column: Views (Avg.), Mean: 2822203.50, Median: 2200000.00, Count: 1001
Column: Likes (Avg.), Mean: 330111.89, Median: 244300.00, Count: 1001
Column: Comments (Avg.), Mean: 2042.38, Median: 1200.00, Count: 1001
Column: Shares (Avg.), Mean: 2499.53, Median: 1200.00, Count: 1001