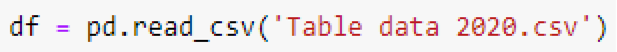
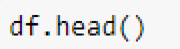
Report

‘Table data 2020.csv’ dataset contains information about activities on YouTube platform from 15th May of 2020 till 26th of September of 2021. It has 20 columns and exactly 500 rows of data. After reading the file and showing first 5 rows with column names,

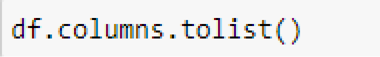


```
df = pd.read_csv('Table data 2020.csv')
```



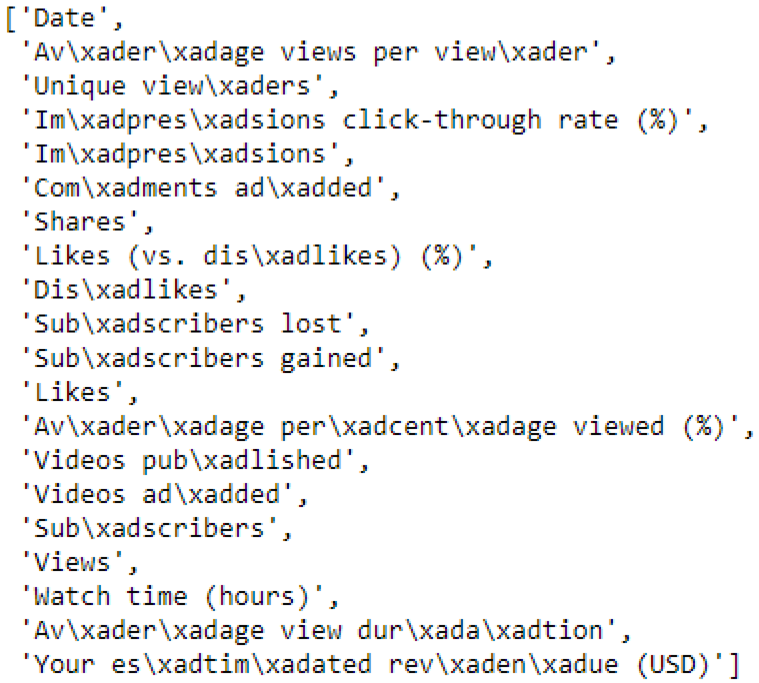
```
df.head()
```

we can see that column names were corrupted. To check this, we used



```
df.columns.tolist()
```

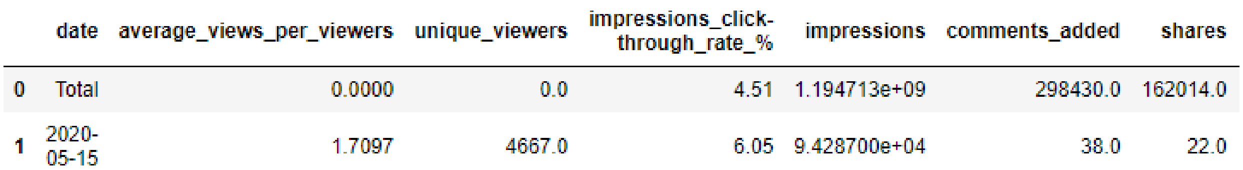
to show a list of the column names. In the following figure, you can notice that we have



```
['Date',
 'Av\xader\xadage views per view\xader',
 'Unique view\xaders',
 'Im\xadpres\xadsions click-through rate (%)',
 'Im\xadpres\xadsions',
 'Com\xadments ad\xadded',
 'Shares',
 'Likes (vs. dis\xadlikes) (%)',
 'Dis\xadlikes',
 'Sub\xadscribers lost',
 'Sub\xadscribers gained',
 'Likes',
 'Av\xader\xadage per\xadcent\xadage viewed (%)',
 'Videos pub\xadlished',
 'Videos ad\xadded',
 'Sub\xadscribers',
 'Views',
 'Watch time (hours)',
 'Av\xader\xadage view dur\xada\xadtion',
 'Your es\xadtim\xadated rev\xaden\xadue (USD)']
```

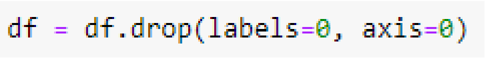
unnecessary \xad in column names. To rename the column names df.rename() was used.

DataFrame had also irrelevant data (total of all column values) which needed to be removed.



	date	average_views_per_viewers	unique_viewers	impressions_click-through_rate_%	impressions	comments_added	shares
0	Total	0.0000	0.0	4.51	1.194713e+09	298430.0	162014.0
1	2020-05-15	1.7097	4667.0	6.05	9.428700e+04	38.0	22.0

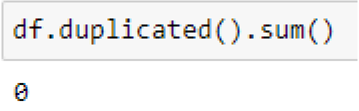
To drop this row, we used where labels=0 represents index and axis=0 row.



```
df = df.drop(labels=0, axis=0)
```

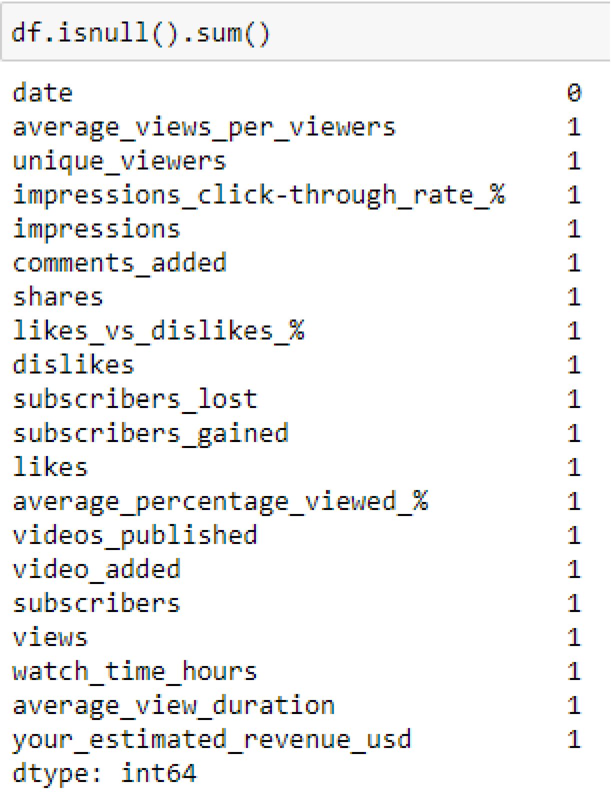
.

In the next steps, we want to check whether dataframe contains duplicates and null values.



```
df.duplicated().sum()
```

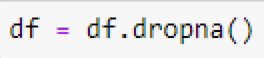
```
0
```



```
df.isnull().sum()
```

```
date                                0
average_views_per_viewers           1
unique_viewers                      1
impressions_click-through_rate_%    1
impressions                         1
comments_added                      1
shares                              1
likes_vs_dislikes_%                 1
dislikes                            1
subscribers_lost                    1
subscribers_gained                  1
likes                               1
average_percentage_viewed_%         1
videos_published                    1
video_added                         1
subscribers                         1
views                               1
watch_time_hours                    1
average_view_duration               1
your_estimated_revenue_usd          1
dtype: int64
```

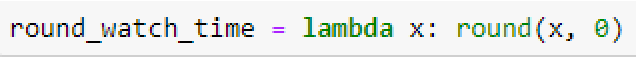
So here, we have some null values, which are recommended to be removed or to apply an imputation (e.g. replacing with mean/median). In our case, we will just drop them by this code:



```
df = df.dropna()
```

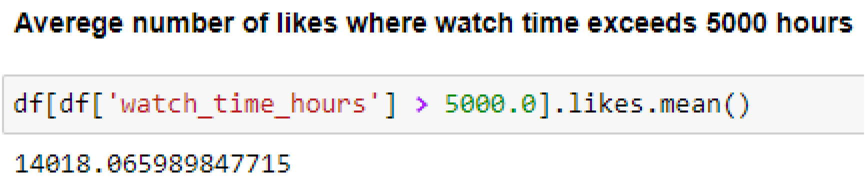
.

To round values in watch_time_hours we can use the lambda function which was defined as follows



```
round_watch_time = lambda x: round(x, 0)
```

For example, 426.8278 will be rounded to 427.0 because the float number was rounded up to 0 decimal.

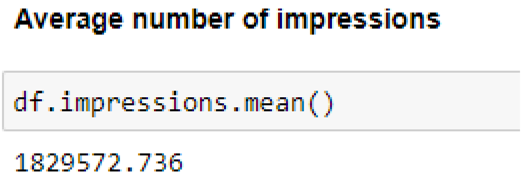


Averege number of likes where watch time exceeds 5000 hours

```
df[df['watch_time_hours'] > 5000.0].likes.mean()
```

```
14018.065989847715
```

Here we have found an average number of likes where watch time is more than 5000 hours. We filtered our dataframe writing condition and accessing to the likes column to find mean().

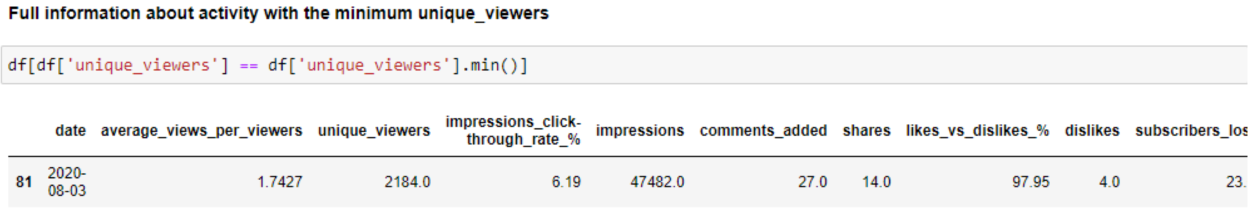


Average number of impressions

```
df.impressions.mean()
```

```
1829572.736
```

In this step, we just calculated average number of all impressions in dataset.

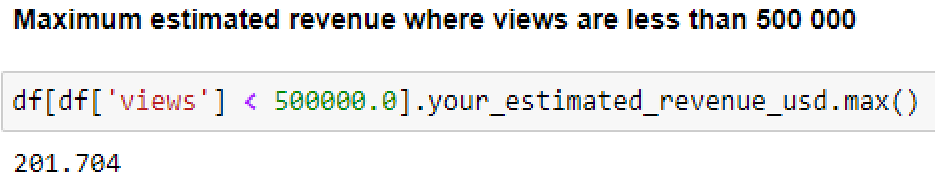


Full information about activity with the minimum unique_viewers

```
df[df['unique_viewers'] == df['unique_viewers'].min()]
```

	date	average_views_per_viewers	unique_viewers	impressions_click-through_rate_%	impressions	comments_added	shares	likes_vs_dislikes_%	dislikes	subscribers_los
81	2020-08-03	1.7427	2184.0	6.19	47482.0	27.0	14.0	97.95	4.0	23.

From this code, we could see the result as a full information about activity with the minimum unique viewers where it is 2184. We can see its number of shares, how many people unsubscribed, number of comments added, and so on.

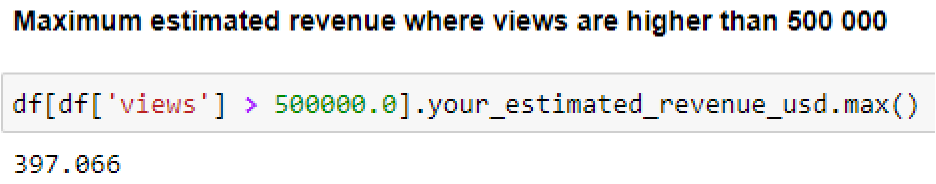


Maximum estimated revenue where views are less than 500 000

```
df[df['views'] < 500000.0].your_estimated_revenue_usd.max()
```

```
201.704
```

We could analyze that a maximum estimated revenue is around $202 where the views are less than 500 000.



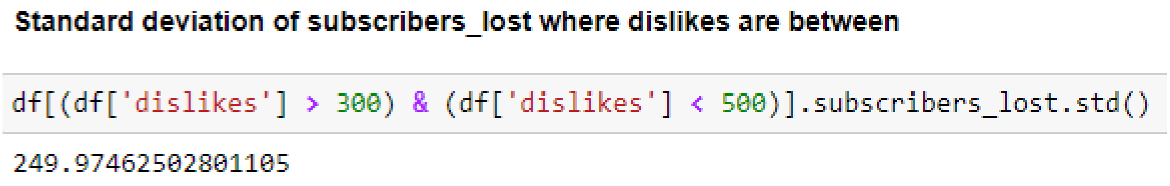
Maximum estimated revenue where views are higher than 500 000

```
df[df['views'] > 500000.0].your_estimated_revenue_usd.max()
```

```
397.066
```

It is the same approach as was in previous, but views are more than 500 000. Maximum estimated revenue is $397.

From these two, we can confirm that the number of views are correlated with the estimated revenue. The more views you get, the more revenue you can expect.

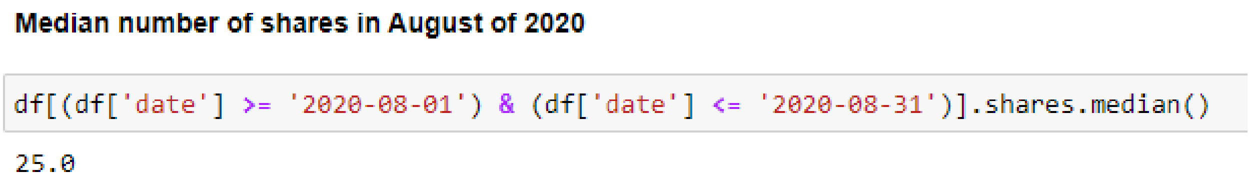


Standard deviation of subscribers_lost where dislikes are between

```
df[(df['dislikes'] > 300) & (df['dislikes'] < 500)].subscribers_lost.std()
```

```
249.97462502801105
```

Probability distribution of subscribers_lost is getting higher whenever the range of dislikes is increased.

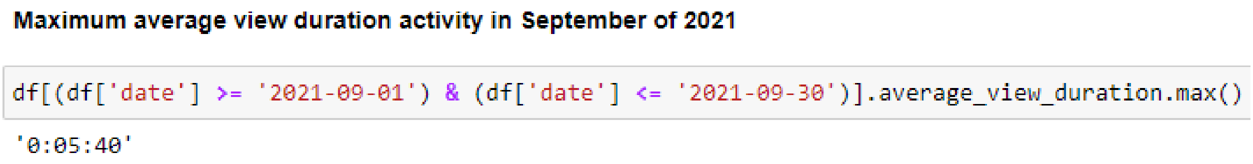


Median number of shares in August of 2020

```
df[(df['date'] >= '2020-08-01') & (df['date'] <= '2020-08-31')].shares.median()
```

```
25.0
```

Median value can be calculated with median() function. Here we computed median number of shares exactly in one month which is August of 2020.



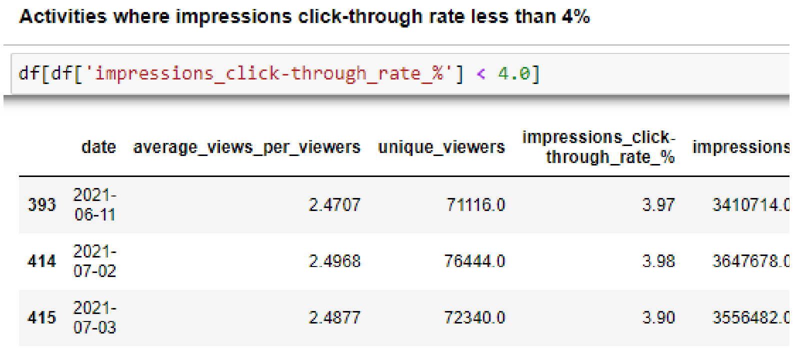
Maximum average view duration activity in September of 2021

```
df[(df['date'] >= '2021-09-01') & (df['date'] <= '2021-09-30')].average_view_duration.max()
```

```
'0:05:40'
```

This code refers to the maximum average view duration in September of 2021. As it shows here, it is almost 6 minutes.

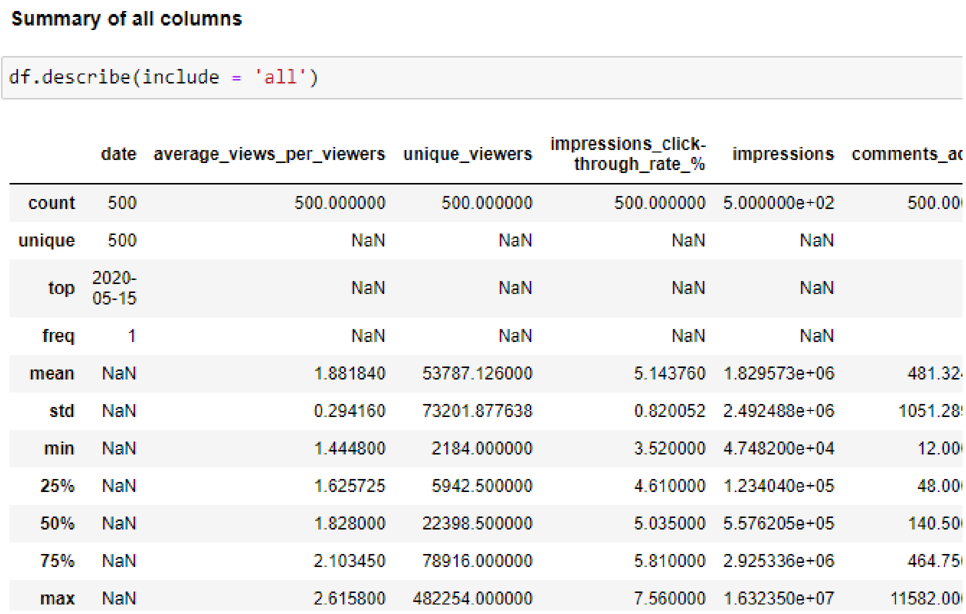
The following figure shows all activities where rate of the click-through impressions is less than 4%.



Activities where impressions click-through rate less than 4%

```
df[df['impressions_click-through_rate_%'] < 4.0]
```

	date	average_views_per_viewers	unique_viewers	impressions_click-through_rate_%	impressions
393	2021-06-11	2.4707	71116.0	3.97	3410714.0
414	2021-07-02	2.4968	76444.0	3.98	3647678.0
415	2021-07-03	2.4877	72340.0	3.90	3556482.0



Summary of all columns

```
df.describe(include = 'all')
```

	date	average_views_per_viewers	unique_viewers	impressions_click-through_rate_%	impressions	comments_ac
count	500	500.000000	500.000000	500.000000	5.000000e+02	500.00
unique	500	NaN	NaN	NaN	NaN	
top	2020-05-15	NaN	NaN	NaN	NaN	
freq	1	NaN	NaN	NaN	NaN	
mean	NaN	1.881840	53787.126000	5.143760	1.829573e+06	481.32
std	NaN	0.294160	73201.877638	0.820052	2.492488e+06	1051.28
min	NaN	1.444800	2184.000000	3.520000	4.748200e+04	12.00
25%	NaN	1.625725	5942.500000	4.610000	1.234040e+05	48.00
50%	NaN	1.828000	22398.500000	5.035000	5.576205e+05	140.50
75%	NaN	2.103450	78916.000000	5.810000	2.925336e+06	464.75
max	NaN	2.615800	482254.000000	7.560000	1.632350e+07	11582.00

This is the summary of all columns where we passed an argument include = ‘all’ to describe() function which means that it includes count, unique, top and freq.