top-5000-youtube-channel-analysis

October 29, 2024

1. Importing Libraries

```
[248]: import pandas as pd
       import numpy as np
       import matplotlib.pyplot as plt
       import seaborn as sns
```

2. Importing the dataset

```
[249]: newdata = pd.read_csv(r"H:\DA. Python\6. 5000 Youtube Channel␣
         ↪Analysis\top-5000-youtube-channels.csv")
```

3. Checking the dataset.

```
[250]: newdata.shape
```

```
[250]: (5000, 6)
```

4. Checking the Top 5 Rows

```
[251]: newdata.head(5)
```

```
[251]:   Rank Grade                 Channel name Video Uploads Subscribers  \
       0  1st   A++                      Zee TV         82757    18752951
       1  2nd   A++                    T-Series         12661    61196302
       2  3rd   A++   Cocomelon - Nursery Rhymes           373    19238251
       3  4th   A++                   SET India         27323    31180559
       4  5th   A++                         WWE         36756    32852346

          Video views
       0  20869786591
       1  47548839843
       2   9793305082
       3  22675948293
       4  26273668433
```

5. Checking the Top 5 Rows

```
[252]: newdata.tail(5)
```

```
[252]:         Rank Grade       Channel name Video Uploads Subscribers  Video views
       4995  4,996th    B+      Uras Benlioğlu           706     2072942    441202795
       4996  4,997th    B+    HI-TECH MUSIC LTD           797     1055091    377331722
       4997  4,998th    B+          Mastersaint           110     3265735    311758426
       4998  4,999th    B+      Bruce McIntosh          3475       32990     14563764
       4999  5,000th    B+           SehatAQUA           254       21172     73312511
```

6. Get Information About The Dataset Like Total Number Of Rows, Total Number Of Columns, Datatypes of Each Column And Memory Requirement

```
[253]: newdata.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5000 entries, 0 to 4999
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Rank           5000 non-null   object
 1   Grade          5000 non-null   object
 2   Channel name   5000 non-null   object
 3   Video Uploads  5000 non-null   object
 4   Subscribers    5000 non-null   object
 5   Video views    5000 non-null   int64
dtypes: int64(1), object(5)
memory usage: 234.5+ KB
```

7. Get Overall Statistics About The Dataframe

```
[254]: newdata.describe(include="all")
```

```
[254]:            Rank Grade           Channel name Video Uploads Subscribers  \
       count       5000  5000                   5000          5000        5000
       unique      5000     6                   4993          2286        4612
       top      5,000th    B+   Learn Colors For Kids            26          --
       freq           1  2956                      2            17         387
       mean         NaN   NaN                    NaN           NaN         NaN
       std          NaN   NaN                    NaN           NaN         NaN
       min          NaN   NaN                    NaN           NaN         NaN
       25%          NaN   NaN                    NaN           NaN         NaN
       50%          NaN   NaN                    NaN           NaN         NaN
       75%          NaN   NaN                    NaN           NaN         NaN
       max          NaN   NaN                    NaN           NaN         NaN

                Video views
       count        5000.00
       unique           NaN
       top              NaN
       freq             NaN
```

```
mean    1071449400.15
std     2003843972.12
min             75.00
25%      186232945.75
50%      482054780.00
75%     1124367826.75
max    47548839843.00
```

[255]:
```
# video views are described in exponential format.
# converting it into decimal format

pd.options.display.float_format = '{:.2f}'.format
```

[256]:
```
newdata.describe()
```

[256]:
```
         Video views
count        5000.00
mean   1071449400.15
std    2003843972.12
min            75.00
25%     186232945.75
50%     482054780.00
75%    1124367826.75
max   47548839843.00
```

8. Data Cleaning (Replace '–' to NaN)

[257]:
```
newdata.head()
```

[257]:
```
  Rank Grade                Channel name Video Uploads Subscribers  \
0  1st   A++                     Zee TV         82757    18752951
1  2nd   A++                   T-Series         12661    61196302
2  3rd   A++  Cocomelon - Nursery Rhymes          373    19238251
3  4th   A++                  SET India         27323    31180559
4  5th   A++                        WWE         36756    32852346

   Video views
0  20869786591
1  47548839843
2   9793305082
3  22675948293
4  26273668433
```

[258]:
```
# Replacing this "--" with "np.nan". So we can drop this particular rows also␣
↪we've to make "regular expression = true"

newdata.replace("--", np.nan, regex=True)
```

```
[258]:          Rank Grade                 Channel name Video Uploads Subscribers  \
       0        1st   A++                       Zee TV         82757    18752951
       1        2nd   A++                     T-Series         12661    61196302
       2        3rd   A++   Cocomelon - Nursery Rhymes           373    19238251
       3        4th   A++                    SET India         27323    31180559
       4        5th   A++                          WWE         36756    32852346
       …          …    …                             …           …           …
       4995 4,996th    B+               Uras Benlioğlu           706     2072942
       4996 4,997th    B+            HI-TECH MUSIC LTD           797     1055091
       4997 4,998th    B+                  Mastersaint           110     3265735
       4998 4,999th    B+               Bruce McIntosh          3475       32990
       4999 5,000th    B+                   SehatAQUA            254       21172

            Video views
       0    20869786591
       1    47548839843
       2     9793305082
       3    22675948293
       4    26273668433
       …              …
       4995   441202795
       4996   377331722
       4997   311758426
       4998    14563764
       4999    73312511

       [5000 rows x 6 columns]
```

9. Checking Null Values In The Dataset

```
[259]: newdata.isnull().sum()
```

```
[259]: Rank             0
       Grade            0
       Channel name     0
       Video Uploads    0
       Subscribers      0
       Video views      0
       dtype: int64
```

```
[260]: # lets find percentage of missing values

       per_missing = newdata.isnull().sum() * 100 / len(newdata)
```

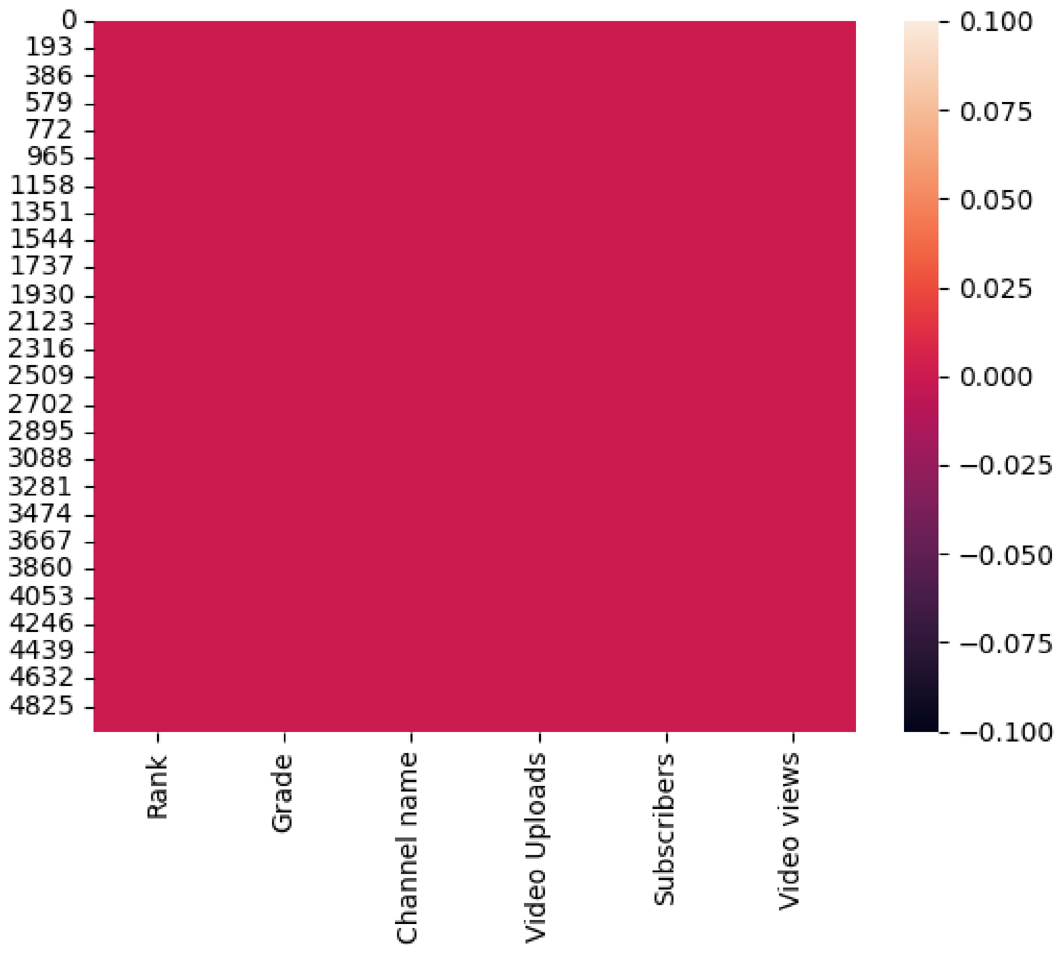
```
[261]: print(per_missing)
```

```
Rank             0.00
Grade            0.00
```

```
Channel name     0.00
Video Uploads    0.00
Subscribers      0.00
Video views      0.00
dtype: float64
```

[262]:
```
sns.heatmap(newdata.isnull())
```

[262]:
```
<Axes: >
```



[263]:
```
newdata.dropna(axis = 0 , inplace = True)

# axis = 0 - because we are dropping values from "rows" which contains missing␣
 ↪values
```

10. Data Cleaning [Rank Column]

[264]:
```
newdata.head(5)
```

```
[264]:   Rank Grade                Channel name Video Uploads Subscribers  \
       0  1st   A++                     Zee TV         82757    18752951
       1  2nd   A++                   T-Series         12661    61196302
       2  3rd   A++  Cocomelon - Nursery Rhymes          373    19238251
       3  4th   A++                  SET India         27323    31180559
       4  5th   A++                        WWE         36756    32852346

          Video views
       0  20869786591
       1  47548839843
       2   9793305082
       3  22675948293
       4  26273668433
```

```
[265]: newdata.dtypes
```

```
[265]: Rank               object
       Grade              object
       Channel name       object
       Video Uploads      object
       Subscribers        object
       Video views         int64
       dtype: object
```

```
[266]: # We are gonna remove the string values from the "Rank" column
```

```
[267]: # At the end we have to give this data to machine learning algorithms.
       # Most of the ML algorithms can only understand numerical values. Either int or␣
        ↪float.
       # So at the end we have to convert all the column to either int or float.

       # to clean this rank column, we are going to perform 3 steps.
       # 1 - we will remove the string in "Rank column"
       # 2 - we will remove this commas
       # 3 - we will convert the data types of rank columns to int
```

```
[268]: newdata.head(2)
```

```
[268]:   Rank Grade Channel name Video Uploads Subscribers  Video views
       0  1st   A++       Zee TV         82757    18752951  20869786591
       1  2nd   A++     T-Series         12661    61196302  47548839843
```

```
[269]: #Here we are using slicing to modify our existing dataframe, and assigning back.
        ↪ ("newdata[Rank]")

       newdata["Rank"] = newdata["Rank"].str[0:-2]
```

[270]:
```
newdata.head()
```

[270]:
```
   Rank Grade                 Channel name Video Uploads Subscribers  \
0    1  A++                         Zee TV         82757    18752951
1    2  A++                       T-Series         12661    61196302
2    3  A++   Cocomelon - Nursery Rhymes           373    19238251
3    4  A++                      SET India         27323    31180559
4    5  A++                            WWE         36756    32852346

   Video views
0  20869786591
1  47548839843
2   9793305082
3  22675948293
4  26273668433
```

[271]:
```
newdata.tail()
```

[271]:
```
        Rank Grade       Channel name Video Uploads Subscribers  Video views
4995  4,996    B+      Uras Benlioğlu           706     2072942    441202795
4996  4,997    B+   HI-TECH MUSIC LTD           797     1055091    377331722
4997  4,998    B+          Mastersaint           110     3265735    311758426
4998  4,999    B+      Bruce McIntosh          3475       32990     14563764
4999  5,000    B+            SehatAQUA           254       21172     73312511
```

[272]:
```
#Removing "," from Ranks

newdata["Rank"] = newdata["Rank"].str.replace(',','')
```

[273]:
```
newdata.tail()
```

[273]:
```
       Rank Grade       Channel name Video Uploads Subscribers  Video views
4995  4996    B+      Uras Benlioğlu           706     2072942    441202795
4996  4997    B+   HI-TECH MUSIC LTD           797     1055091    377331722
4997  4998    B+          Mastersaint           110     3265735    311758426
4998  4999    B+      Bruce McIntosh          3475       32990     14563764
4999  5000    B+            SehatAQUA           254       21172     73312511
```

[274]:
```
newdata.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5000 entries, 0 to 4999
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Rank           5000 non-null   object
 1   Grade          5000 non-null   object
 2   Channel name   5000 non-null   object
 3   Video Uploads  5000 non-null   object
 4   Subscribers    5000 non-null   object
 5   Video views    5000 non-null   int64
dtypes: int64(1), object(5)
memory usage: 234.5+ KB
```

[275]:
```
#Convert data type of Rank from OBJ to INT

newdata['Rank'] = newdata['Rank'].str.replace(",","").astype("int")
```

[276]:
```
newdata.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5000 entries, 0 to 4999
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Rank           5000 non-null   int64
 1   Grade          5000 non-null   object
 2   Channel name   5000 non-null   object
 3   Video Uploads  5000 non-null   object
 4   Subscribers    5000 non-null   object
 5   Video views    5000 non-null   int64
dtypes: int64(2), object(4)
memory usage: 234.5+ KB
```

11. Data Cleaning [Video Uploads & Subscribers]

[277]:
```
newdata.head(5)
```

[277]:
```
   Rank Grade                Channel name Video Uploads Subscribers  \
0     1   A++                      Zee TV         82757    18752951
1     2   A++                    T-Series         12661    61196302
2     3   A++   Cocomelon - Nursery Rhymes          373    19238251
3     4   A++                   SET India         27323    31180559
4     5   A++                         WWE         36756    32852346

   Video views
0  20869786591
1  47548839843
2   9793305082
3  22675948293
4  26273668433
```

[278]:
```
newdata.info()
```

```
<class 'pandas.core.frame.DataFrame'>
```

```
RangeIndex: 5000 entries, 0 to 4999
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype 
---  ------         --------------  ----- 
 0   Rank           5000 non-null   int64 
 1   Grade          5000 non-null   object
 2   Channel name   5000 non-null   object
 3   Video Uploads  5000 non-null   object
 4   Subscribers    5000 non-null   object
 5   Video views    5000 non-null   int64 
dtypes: int64(2), object(4)
memory usage: 234.5+ KB
```

[279]:
```
#Cleaning the data, replacing "--" with "nan"
newdata['Video Uploads'] = newdata['Video Uploads'].replace("--", np.nan)

#Converting data type from OBJ to INT
newdata['Video Uploads'] = newdata['Video Uploads'].astype("float").
 ↪astype("Int64")
```

[280]:
```
newdata.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5000 entries, 0 to 4999
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype 
---  ------         --------------  ----- 
 0   Rank           5000 non-null   int64 
 1   Grade          5000 non-null   object
 2   Channel name   5000 non-null   object
 3   Video Uploads  4994 non-null   Int64 
 4   Subscribers    5000 non-null   object
 5   Video views    5000 non-null   int64 
dtypes: Int64(1), int64(2), object(3)
memory usage: 239.4+ KB
```

[281]:
```
#Replace NAN with 0
newdata['Video Uploads'] = newdata['Video Uploads'].fillna(0)

#Convert the column to integer
newdata['Video Uploads'] = newdata['Video Uploads'].astype('int')
```

[282]:
```
newdata.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5000 entries, 0 to 4999
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype 
```

```
---  ------         --------------  -----
 0   Rank           5000 non-null   int64
 1   Grade          5000 non-null   object
 2   Channel name   5000 non-null   object
 3   Video Uploads  5000 non-null   int64
 4   Subscribers    5000 non-null   object
 5   Video views    5000 non-null   int64
dtypes: int64(3), object(3)
memory usage: 234.5+ KB
```

[283]:
```
newdata.dtypes
```

[283]:
```
Rank              int64
Grade            object
Channel name     object
Video Uploads     int64
Subscribers      object
Video views       int64
dtype: object
```

[284]:
```
#Replace -- with nan

newdata['Subscribers'] = newdata['Subscribers'].replace("--", np.nan)
```

[285]:
```
#Replace nan to 0

newdata['Subscribers'] = newdata['Subscribers'].fillna(0)
```

[286]:
```
#Replace non numeric value with NaN
newdata['Subscribers'] = pd.to_numeric(newdata['Subscribers'], errors =␣
 ↪"coerce" )

#Convert the column to integer
newdata['Subscribers'] = newdata['Subscribers'].astype("Int32")
```

[287]:
```
newdata.dtypes
```

[287]:
```
Rank              int64
Grade            object
Channel name     object
Video Uploads     int64
Subscribers       Int32
Video views       int64
dtype: object
```

12. Data Cleaning [Grade Column]

[288]:
```
newdata.head()
```

[288]:
```
   Rank Grade                Channel name  Video Uploads  Subscribers  \
0     1  A++                       Zee TV          82757     18752951
1     2  A++                     T-Series          12661     61196302
2     3  A++   Cocomelon - Nursery Rhymes            373     19238251
3     4  A++                    SET India          27323     31180559
4     5  A++                          WWE          36756     32852346

   Video views
0  20869786591
1  47548839843
2   9793305082
3  22675948293
4  26273668433
```

[289]:
```
#Finding the unique values
newdata['Grade'].unique()
```

[289]:
```
array(['A++ ', 'A+ ', 'A ', '\xa0 ', 'A- ', 'B+ '], dtype=object)
```

[290]:
```
#Assigning values to Numeric values
newdata['Grade'] = newdata['Grade'].map({'A++ ':5, 'A+ ':4, 'A ':3, 'A- ':2,␣
 ↪'B+ ':1})
```

[291]:
```
newdata.head()
```

[291]:
```
   Rank  Grade                Channel name  Video Uploads  Subscribers  \
0     1   5.00                      Zee TV          82757     18752951
1     2   5.00                    T-Series          12661     61196302
2     3   5.00  Cocomelon - Nursery Rhymes            373     19238251
3     4   5.00                   SET India          27323     31180559
4     5   5.00                         WWE          36756     32852346

   Video views
0  20869786591
1  47548839843
2   9793305082
3  22675948293
4  26273668433
```

[292]:
```
newdata.dtypes
```

[292]:
```
Rank                int64
Grade             float64
Channel name       object
Video Uploads       int64
Subscribers         Int32
Video views         int64
```

```
dtype: object
```

13. Find Average Views For Each Channel

```
[293]: newdata.columns
```

```
[293]: Index(['Rank', 'Grade', 'Channel name', 'Video Uploads', 'Subscribers',
              'Video views'],
             dtype='object')
```

```
[294]: newdata['avg_views'] = newdata['Video views'] / newdata['Video Uploads']
```

```
[295]: newdata.head()
```

```
[295]:    Rank  Grade                 Channel name  Video Uploads  Subscribers  \
       0     1   5.00                       Zee TV          82757     18752951
       1     2   5.00                     T-Series          12661     61196302
       2     3   5.00  Cocomelon - Nursery Rhymes            373     19238251
       3     4   5.00                    SET India          27323     31180559
       4     5   5.00                          WWE          36756     32852346

          Video views   avg_views
       0  20869786591   252181.53
       1  47548839843  3755535.89
       2   9793305082 26255509.60
       3  22675948293   829921.62
       4  26273668433   714813.05
```

14. Find Out Top Five Channels With Maximum Number of Video Uploads

```
[296]: newdata.columns
```

```
[296]: Index(['Rank', 'Grade', 'Channel name', 'Video Uploads', 'Subscribers',
              'Video views', 'avg_views'],
             dtype='object')
```

```
[297]: newdata.sort_values(by = "Video Uploads", ascending = True).head()
```

```
[297]:       Rank  Grade                 Channel name  Video Uploads  Subscribers  \
       3072  3073    NaN  Boram Tube Toy Shcool [  …              0       726527
       2323  2324    NaN                       Random              0        12275
       267    268    NaN              MidnightXChannel              0         <NA>
       517    518    NaN                Dusama Pets TV              0         <NA>
       4898  4899    NaN                 ExzoticSlice              0        99785

             Video views  avg_views
       3072    205555289        inf
       2323     17897584        inf
```

```
267      190256974         inf
517       91601494         inf
4898       9745292         inf
```

15. Find Correlation Matrix

[298]:
```
newdata.dtypes
```

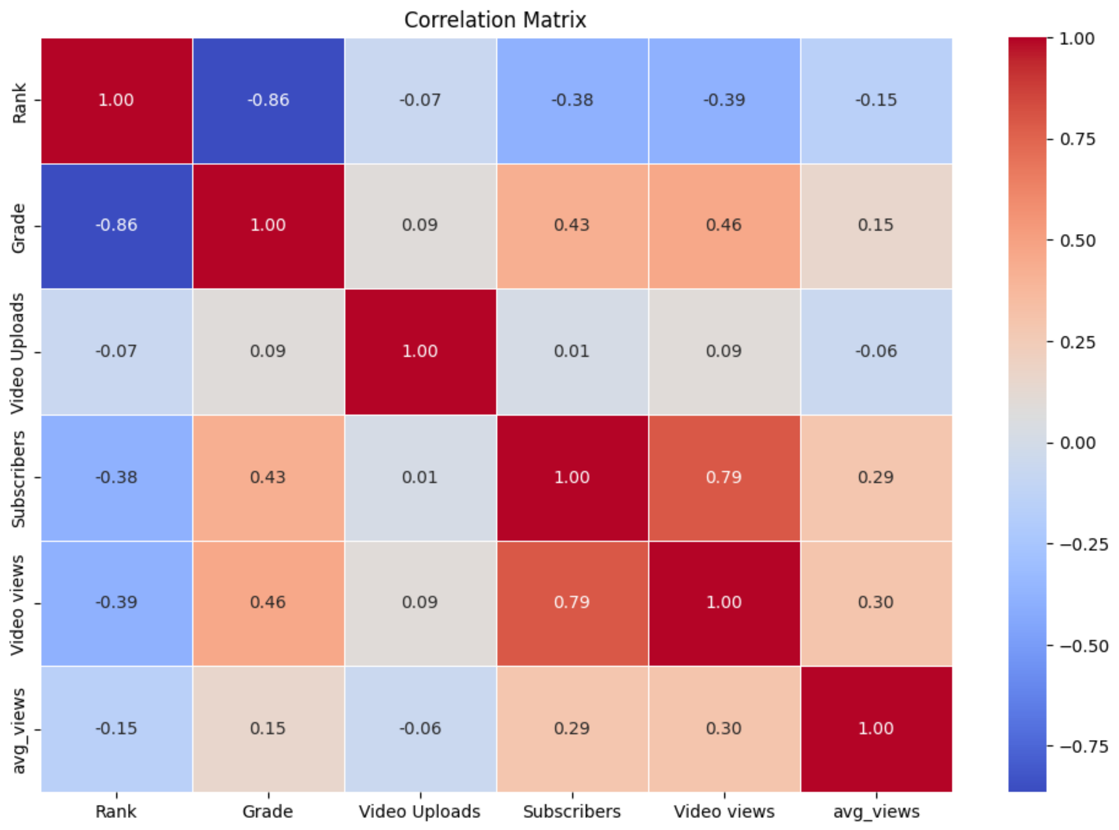
[298]:
```
Rank                 int64
Grade              float64
Channel name        object
Video Uploads        int64
Subscribers          Int32
Video views          int64
avg_views          float64
dtype: object
```

[299]:
```
#Select only numeric values
numeric_column = newdata.select_dtypes(include=[np.number])

#Calculate the corelation matrix
correlation_matrix = numeric_column.corr()
```

[300]:
```
#Creating a heatmap for the Corelation Matrix

plt.figure(figsize=(12,8))
sns.heatmap(correlation_matrix, annot = True, cmap = 'coolwarm', fmt = '.2f',␣
 ↪linewidths = .5)
plt.title("Correlation Matrix")
plt.show()
```



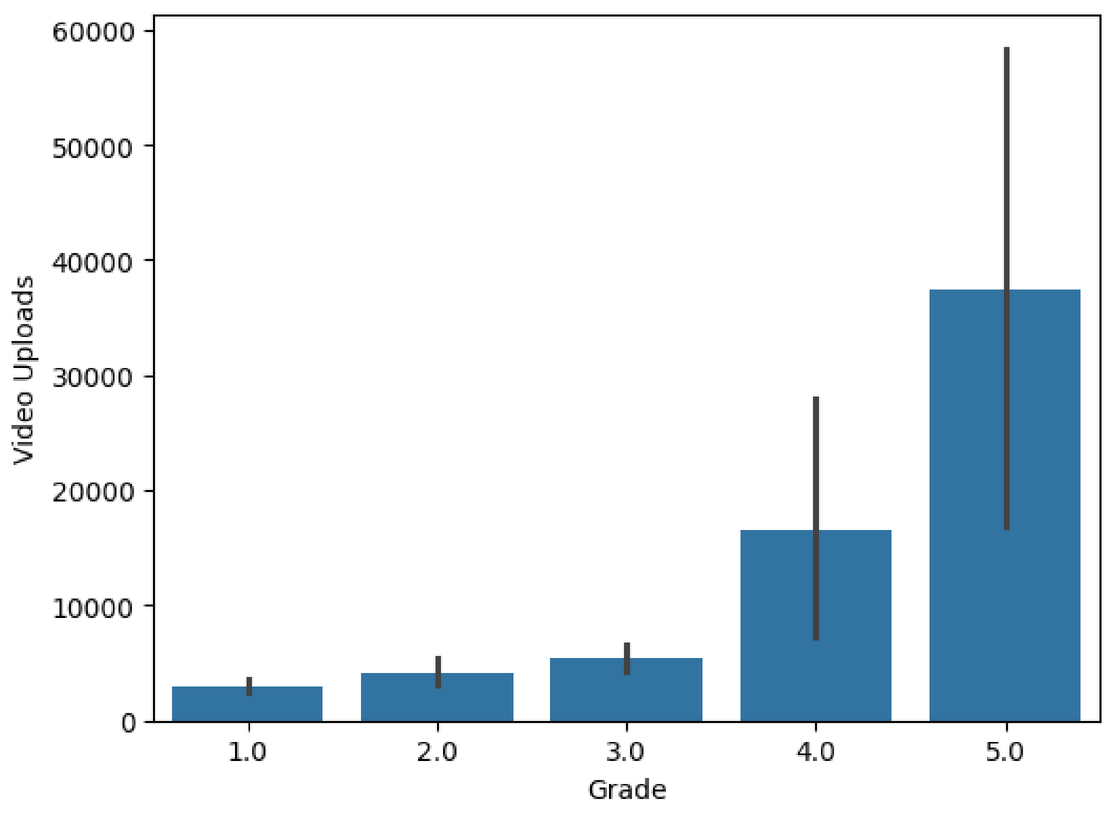
16. Which Grade Has A Maximum Number of Video Uploads?

```
[301]: newdata.columns
```

```
[301]: Index(['Rank', 'Grade', 'Channel name', 'Video Uploads', 'Subscribers',
              'Video views', 'avg_views'],
             dtype='object')
```

```
[302]: sns.barplot(x = "Grade", y = "Video Uploads", data = newdata)
```

```
[302]: <Axes: xlabel='Grade', ylabel='Video Uploads'>
```

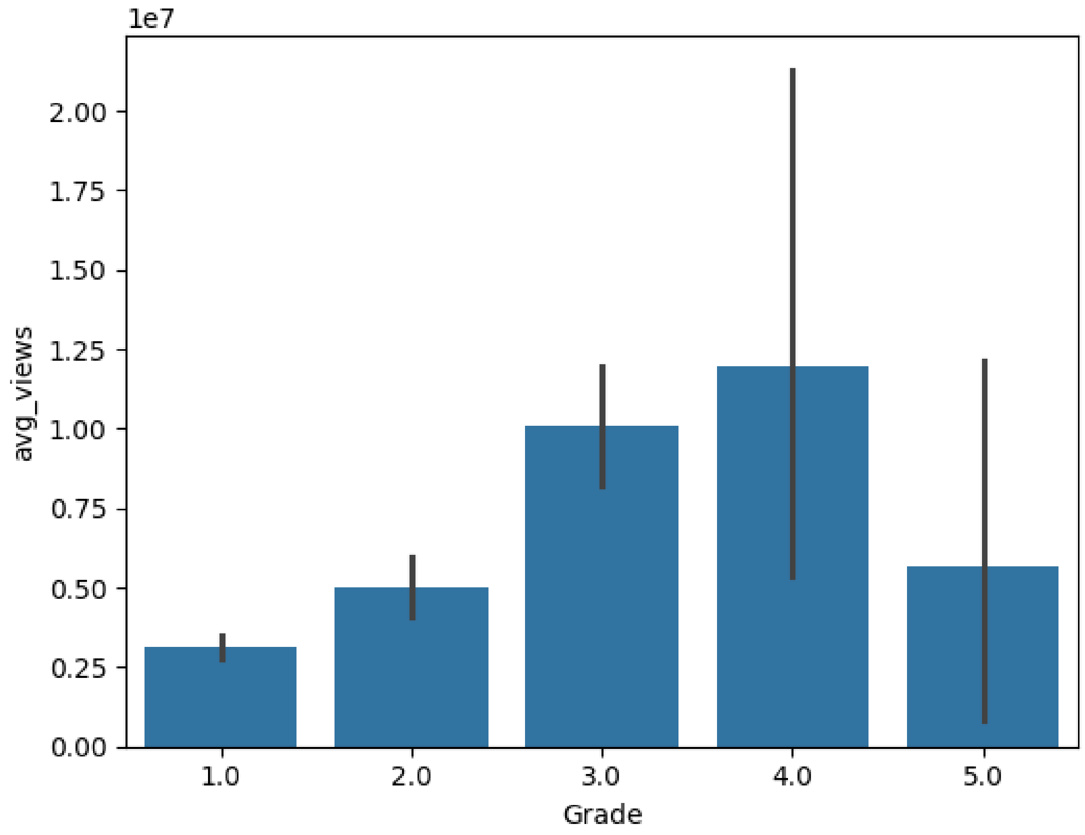


CONCLUSION = From the above graph it is clear that when the grade is high, video upload is also high. Video Uploads are high in 'A++' video channels.

17. Which Grade Has The Highest Average Views?

[303]:
```
sns.barplot(x = 'Grade', y = 'avg_views', data = newdata)
```

[303]:
```
<Axes: xlabel='Grade', ylabel='avg_views'>
```



CONCLUSION = Channels with 'A+' grade has higher number of average views compared to others.

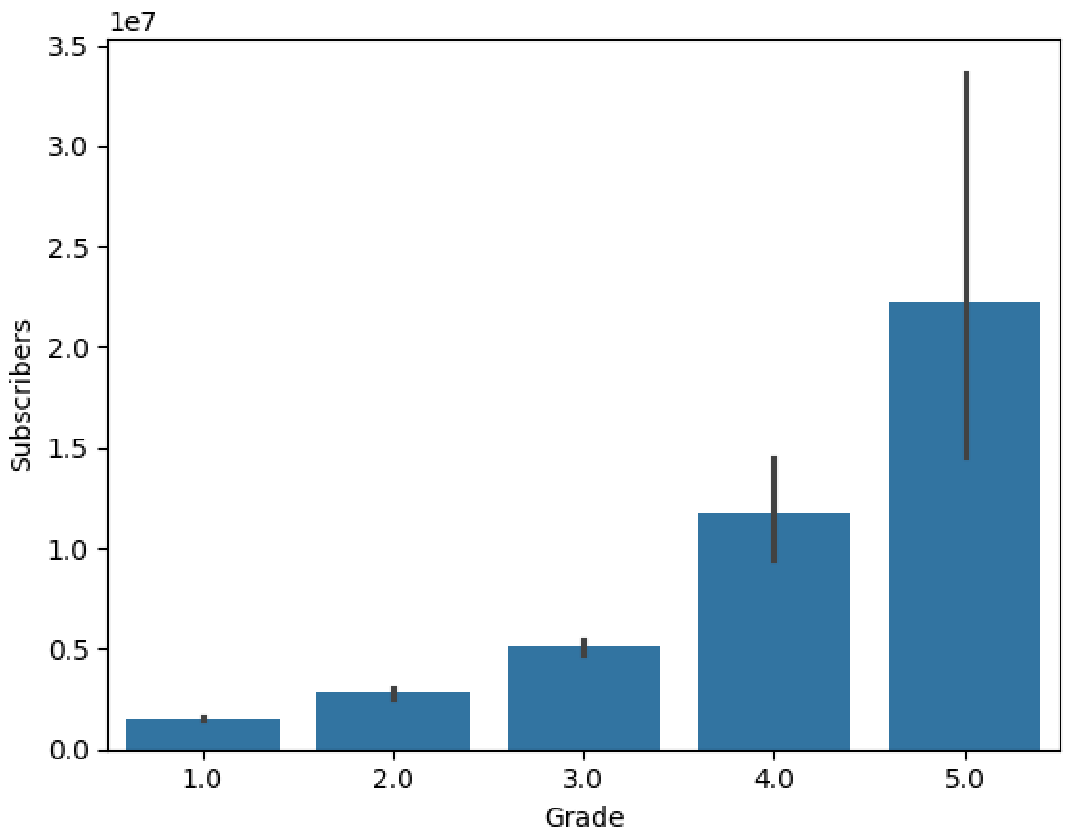
18. Which Grade Has The Highest Number of Subscribers?

```
[304]: newdata.columns
```

```
[304]: Index(['Rank', 'Grade', 'Channel name', 'Video Uploads', 'Subscribers',
              'Video views', 'avg_views'],
             dtype='object')
```

```
[305]: sns.barplot(x = 'Grade', y = 'Subscribers', data = newdata)
```

```
[305]: <Axes: xlabel='Grade', ylabel='Subscribers'>
```



CONCLUSION = From the above graph it is clear that channel with the 'A++' Grade has the highest number of subscribers

21. Which Grade Has The Highest Video Views?

[306]:
```
newdata.columns
```

[306]:
```
Index(['Rank', 'Grade', 'Channel name', 'Video Uploads', 'Subscribers',
       'Video views', 'avg_views'],
      dtype='object')
```

[307]:
```
newdata.groupby('Grade')['Video views'].sum()
```

[307]:
```
Grade
1.00    1556398001373
2.00    1066136831368
3.00    2273948590311
4.00     248177945463
5.00     211990911928
Name: Video views, dtype: int64
```