**Name:** Shrey Pendurkar
**Class:** D15C
**Batch:** C
**Roll No:** 64

# DMBI - Experiment 3

**Aim: To perform exploratory data analysis and visualization on the dataset using python.**

## Theory:

Exploratory Data Analysis (EDA) is the process of exploring, summarizing, and visualizing data to understand its main characteristics before applying statistical models or predictive analytics. For the **Global YouTube Statistics 2023** dataset (top ~1000 YouTubers worldwide), EDA helps uncover patterns in subscribers, views, uploads, categories, countries, and earnings.

### 1. Descriptive Statistics

- Numerical summaries (mean, median, variance, min/max, std) are computed for key features like `subscribers`, `video_views`, `uploads`, and `earnings`.
- Helps detect skewness — e.g., subscriber counts are highly **right-skewed** with a few mega-channels dominating.
- Outliers (channels with extremely high views or uploads) are identified, showing the power-law nature of YouTube popularity.

### 2. Target Variable Analysis

- Since this is not a binary classification problem, the **main dependent variables** can be considered as `subscribers` or `video_views`.
- Visualizing top 10 YouTubers by subscribers and by views reveals dominance of entertainment, music, and kids' content.
- Distributions show that most channels have fewer subscribers/views compared to a handful of global leaders.

**3. Correlation Analysis**

- Pearson correlation is calculated between numeric features (`subscribers`, `video_views`, `uploads`, `earnings`).
- Heatmap highlights strong positive correlation between **subscribers and views**, confirming loyal audiences drive viewership.
- Uploads show weaker correlation with subscribers/views, indicating that **quantity doesn't guarantee popularity**.
- Earnings are moderately correlated with views but influenced by other factors (CPM, content type).

**4. Feature Distribution Analysis**

- Histograms/KDE plots reveal that features like `subscribers` and `video_views` follow a **long-tailed distribution** (few channels dominate).
- Boxplots of `views per subscriber` highlight engagement efficiency — some categories deliver disproportionately high views relative to their subscriber base.
- `Uploads` distribution shows many small creators with large video counts but relatively fewer views.

**5. Categorical Feature Analysis**

- Countplots of `category` show **Music, Entertainment, People & Blogs, and Shows** as dominant genres.
- Bar plots by `country` reveal the U.S. and India as leaders, followed by Brazil and other high-population nations.
- Category vs. subscribers/earnings analysis shows that **Show and Music channels** achieve higher engagement than other genres.

**6. Multivariate Visualization**

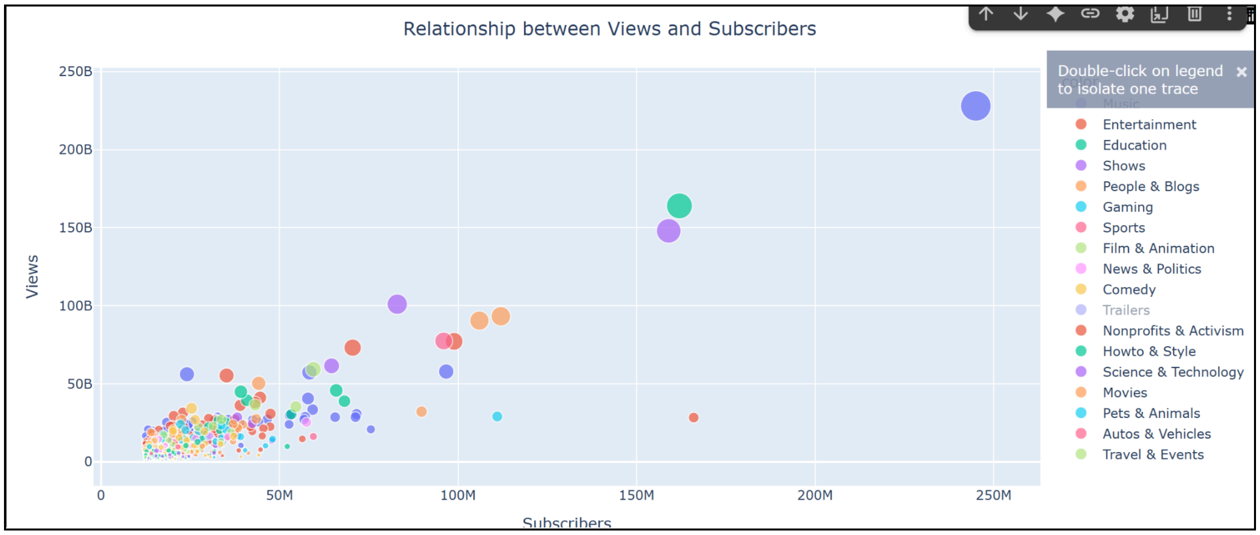
- Scatter plots (`subscribers vs. video_views`) show clusters of highly correlated channels, with outliers representing viral mega-channels.
- Pairplots across (`subscribers`, `views`, `uploads`, `earnings`) highlight differences across categories.
- Grouped boxplots (`category vs. views per subscriber`) illustrate niches where content achieves stronger audience retention.

## Conclusion:

The Exploratory Data Analysis (EDA) and visualization of the Global YouTube Statistics 2023 dataset revealed key insights about top YouTube channels worldwide. Subscribers and video views showed a strong positive correlation, indicating audience size drives engagement, while uploads had a weak influence, emphasizing content quality over quantity. Popular categories like Music and Shows achieved higher engagement and earnings, and countries like the U.S. and India led channel counts and viewership. Visualizations such as scatter plots, pie charts, bar charts, box plots, and correlation heatmaps effectively summarized these relationships.

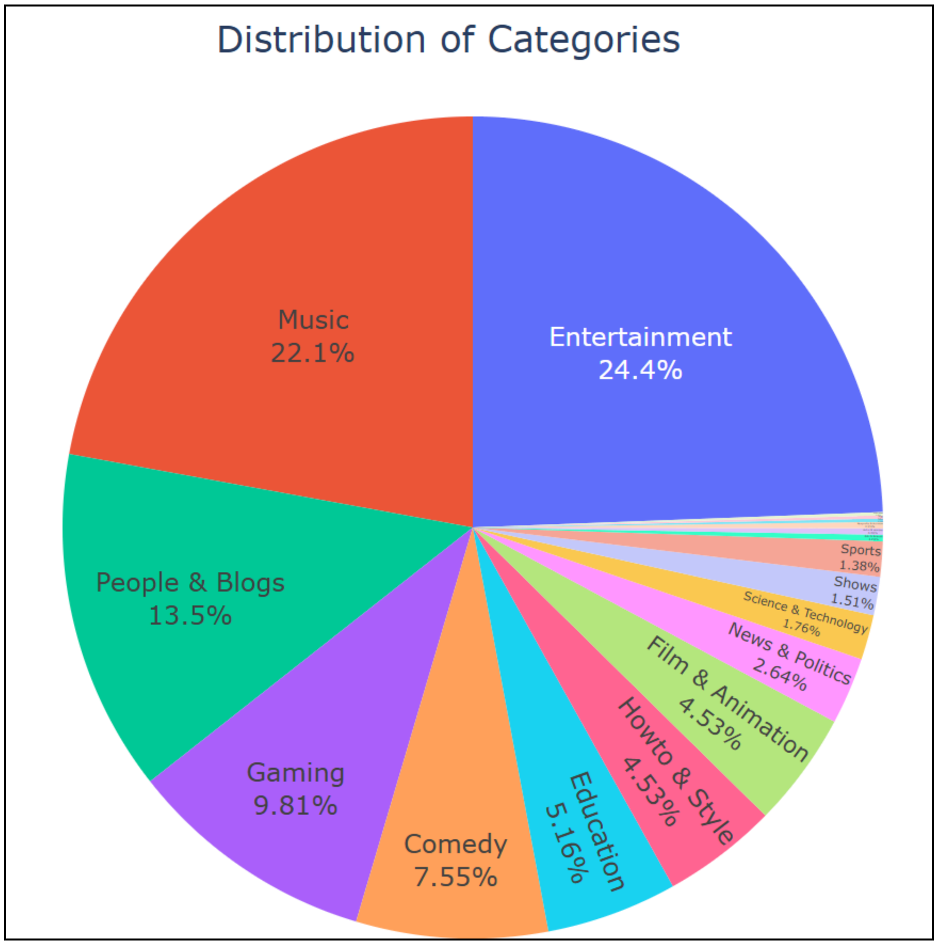
### 1) Scatter Plot:

```
# Creating a scatter plot using plotly express (px)
fig1 = px.scatter(
    x=df["subscribers"],
    y=df["video views"],
    color=df["category"],
    size=df["video views"],
    hover_name=df["Title"]
)
# Updating the layout of the plot
fig1.update_layout(
    xaxis=dict(title="Subscribers"),
    yaxis=dict(title="Views"),
    title="Relationship between Views and Subscribers",
    title_x=0.48
)
# Displaying the plot
fig1.show()
```



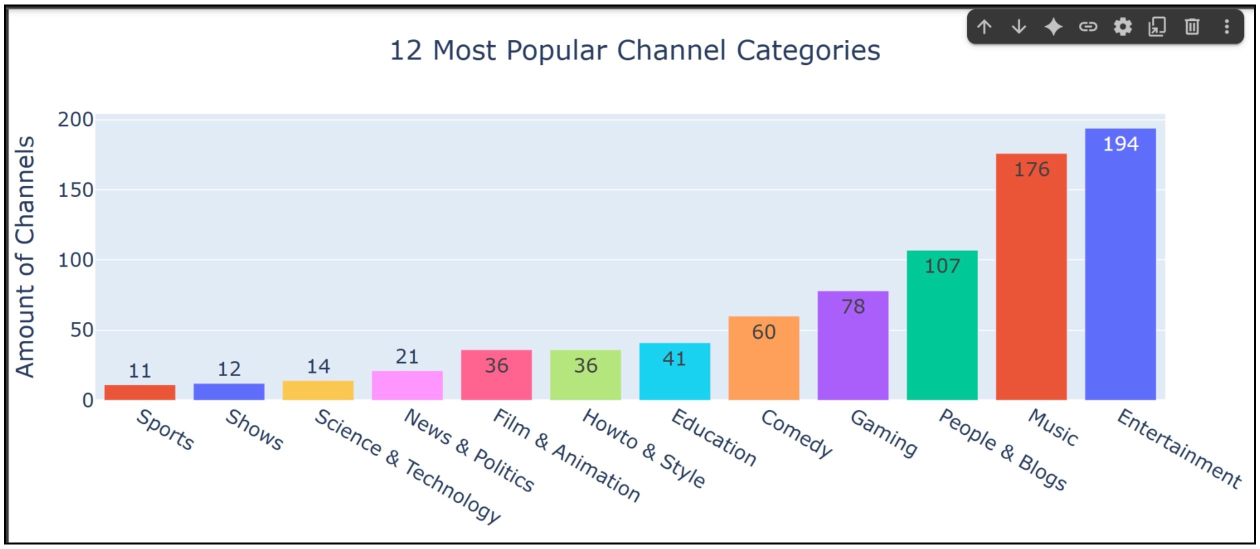
## 2) Pie Chart:

```
# Calculating the count of each category and sorting them in ascending order
category = df['category'].value_counts().sort_values(ascending=True)
# Creating a pie chart using plotly express (px)
fig3 = px.pie(
    values=category.values,
    names=category.index
)
# Updating trace properties for the pie chart
fig3.update_traces(
    textposition='inside',
    textinfo='percent+label'
)
# Updating the layout of the pie chart
fig3.update_layout(
    title_text="Distribution of Categories",
    title_x=0.49,
    uniformtext_minsize=10,
    showlegend=False
)
# Displaying the pie chart
fig3.show()
```



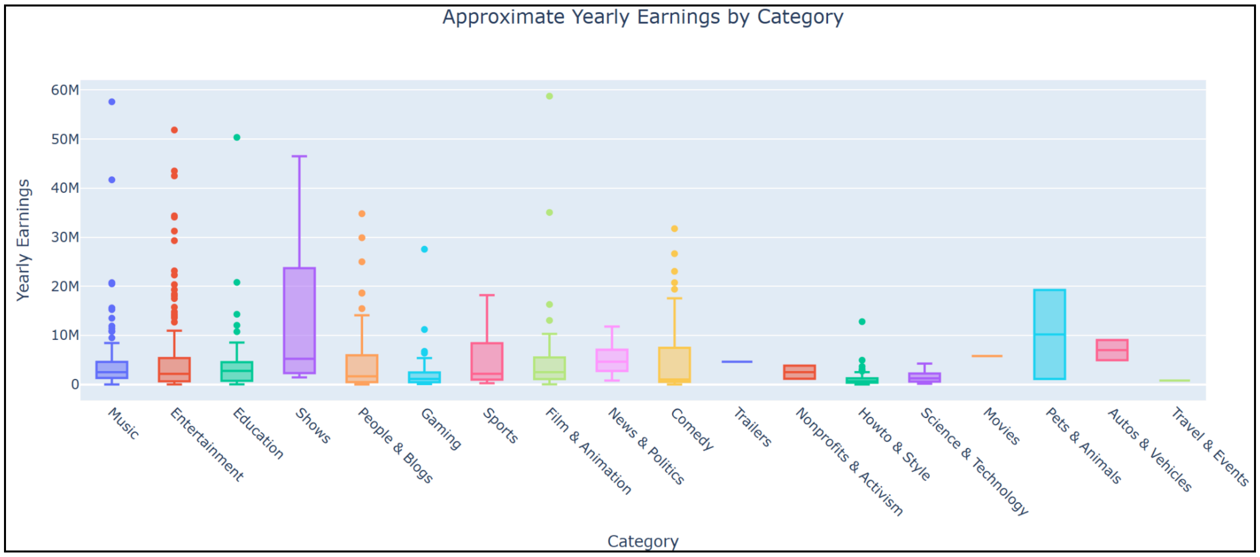
**3) Bar Chart:**

```
# Get the top 12 channel categories
top12_channel_categories = df['category'].value_counts().head(12)
# Plotting the bar chart using plotly.express
fig4 = px.bar(
    x=top12_channel_categories.index,
    y=top12_channel_categories.values,
    color=top12_channel_categories.index,
    text=top12_channel_categories.values,
    title='12 Most Popular Channel Categories',
    labels={'x': 'Category', 'y': 'Amount of Channels'}
)
# Customize the layout
fig4.update_layout(
    title_x=0.5,
    font=dict(size=18),
    showlegend=False,
    xaxis={'categoryorder': 'total ascending'}
)
# Display the plot
fig4.show()
```



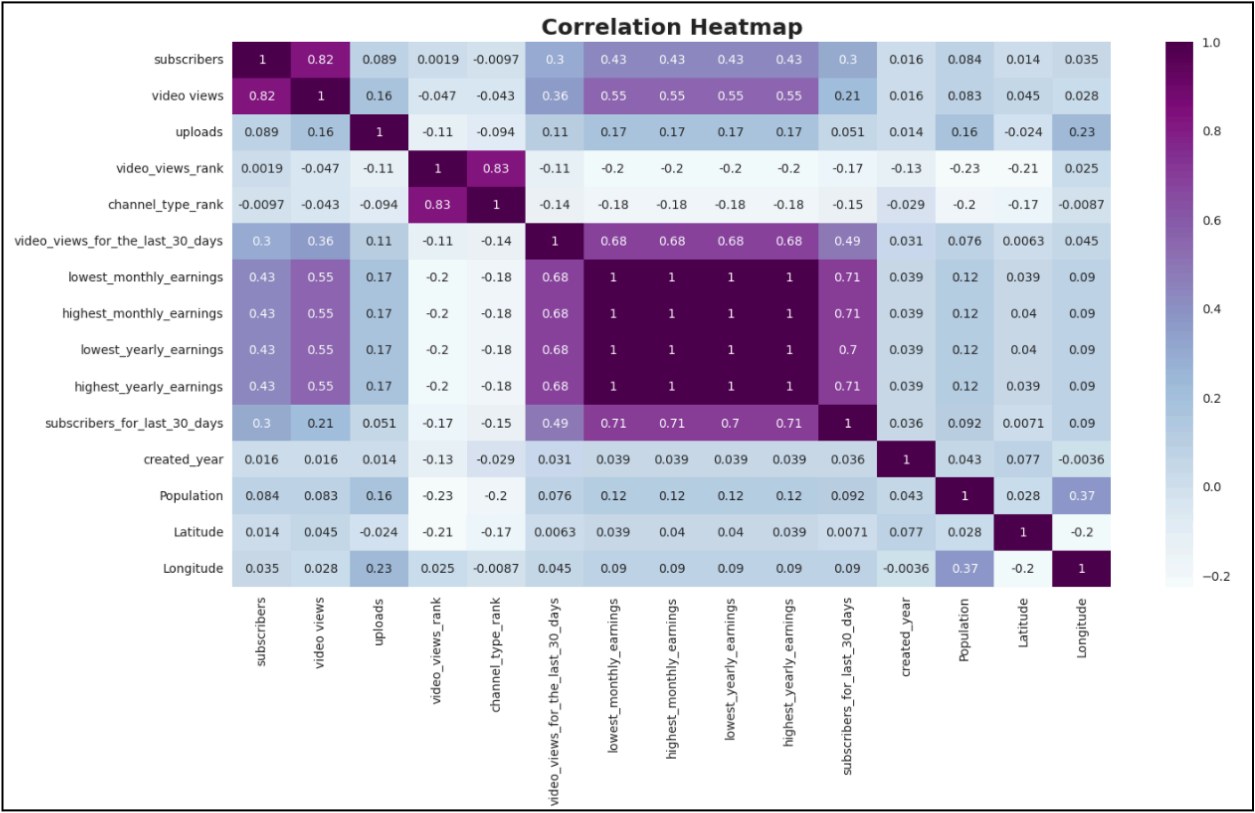
## 4) Box Plot:

```
# Create a box plot using plotly.express
fig7 = px.box(df, x='category', y='yearly_earnings', color='category',
             labels={'yearly_earnings': 'Yearly Earnings', 'category': 'Category'},
             title='Approximate Yearly Earnings by Category')
# Center the title above the bar chart
fig7.update_layout(title=dict(x=0.5), showlegend=False)
# Customize the layout for better readability
fig7.update_layout(xaxis=dict(tickangle=45))
# Show the plot
fig7.show()
```



## 5) Correlation Heatmap:

```
import matplotlib.pyplot as plt
import seaborn as sns
plt.style.use("seaborn")
plt.rcParams['figure.figsize'] = (16, 8)
# Filter numeric columns only
numeric_df = df.select_dtypes(include=[float, int])
# Compute the correlation matrix
correlation_matrix = numeric_df.corr()
# Plot the heatmap
title = "Correlation Heatmap"
plt.title(title, fontsize=18, weight='bold')
sns.heatmap(correlation_matrix, cmap="BuPu", annot=True)
plt.show()
```


Correlation Heatmap
subscribers 1 0.82 0.089 0.0019 -0.0097 0.3 0.43 0.43 0.43 0.43 0.3 0.016 0.084 0.014 0.035
video views 0.82 1 0.16 -0.047 -0.043 0.36 0.55 0.55 0.55 0.55 0.21 0.016 0.083 0.045 0.028
uploads 0.089 0.16 1 -0.11 -0.094 0.11 0.17 0.17 0.17 0.17 0.051 0.014 0.16 -0.024 0.23
video_views_rank 0.0019 -0.047 -0.11 1 0.83 -0.11 -0.2 -0.2 -0.2 -0.2 -0.17 -0.13 -0.23 -0.21 0.025
channel_type_rank -0.0097 -0.043 -0.094 0.83 1 -0.14 -0.18 -0.18 -0.18 -0.18 -0.15 -0.029 -0.2 -0.17 -0.0087
video_views_for_the_last_30_days 0.3 0.36 0.11 -0.11 -0.14 1 0.68 0.68 0.68 0.68 0.49 0.031 0.076 0.0063 0.045
lowest_monthly_earnings 0.43 0.55 0.17 -0.2 -0.18 0.68 1 1 1 1 0.71 0.039 0.12 0.039 0.09
highest_monthly_earnings 0.43 0.55 0.17 -0.2 -0.18 0.68 1 1 1 1 0.71 0.039 0.12 0.04 0.09
lowest_yearly_earnings 0.43 0.55 0.17 -0.2 -0.18 0.68 1 1 1 1 0.7 0.039 0.12 0.04 0.09
highest_yearly_earnings 0.43 0.55 0.17 -0.2 -0.18 0.68 1 1 1 1 0.71 0.039 0.12 0.039 0.09
subscribers_for_last_30_days 0.3 0.21 0.051 -0.17 -0.15 0.49 0.71 0.71 0.7 0.71 1 0.036 0.092 0.0071 0.09
created_year 0.016 0.016 0.014 -0.13 -0.029 0.031 0.039 0.039 0.039 0.039 0.036 1 0.043 0.077 -0.0036
Population 0.084 0.083 0.16 -0.23 -0.2 0.076 0.12 0.12 0.12 0.12 0.092 0.043 1 0.028 0.37
Latitude 0.014 0.045 -0.024 -0.21 -0.17 0.0063 0.039 0.04 0.04 0.039 0.0071 0.077 0.028 1 -0.2
Longitude 0.035 0.028 0.23 0.025 -0.0087 0.045 0.09 0.09 0.09 0.09 0.09 -0.0036 0.37 -0.2 1
subscribers
video views
uploads
video_views_rank
channel_type_rank
video_views_for_the_last_30_days
lowest_monthly_earnings
highest_monthly_earnings
lowest_yearly_earnings
highest_yearly_earnings
subscribers_for_last_30_days
created_year
Population
Latitude
Longitude