ScreenSense: Kids' Screentime Visualization

Week1 and Week 2:

1. **Problem Statement:**

   Analyze kids' screentime patterns to uncover trends by age, gender, location type (urban/rural), device type, day-of-week, and activity category using data visualization. The goal is to present clear, actionable insights for parents, educators, and policymakers.

2. **Project Goal and Workflow:**

   • Data Understanding and Preparation: Comprehend and preprocess the screentime dataset to ensure it is ready for comprehensive analysis

   • Pattern Analysis: Investigate usage trends and patterns across different time periods (weekdays vs weekends), various device types, and diverse activity categories

   • Data Visualization: Create visual representations of key metrics utilizing multiple chart types including bar charts, distribution plots, heatmaps, and comparative visualizations

   • Insight Communication: Develop a visual report or dashboard that effectively summarizes findings and insights for stakeholders without technical backgrounds

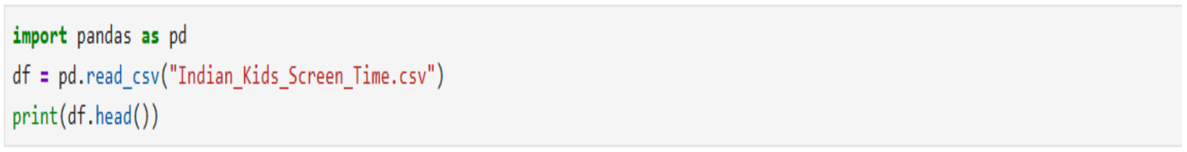
   • Final Deliverable: Prepare and deliver a comprehensive presentation showcasing primary discoveries supported by appropriate visual evidence

   • Workflow: Data Acquisition → Data Preprocessing → Data Analysis → Visual Report →Final Presentation

3. **Dataset Loading:**

   • Download the dataset from Kaggle — Indian Kids Screentime 2025

   • Import the dataset and verify successful loading



```
import pandas as pd
df = pd.read_csv("Indian_Kids_Screen_Time.csv")
print(df.head())
```

   • Dataset successfully loaded into a Pandas DataFrame for inspection.

4. **Explore Dataset:**

   • Examine the structure and completeness of data.

Used

```
df.info()
```
to check column datatypes

```
df.isnull().sum()
```
to find missing values

```
df.shape
```
to verify shape of data

5. Handle missing values and inconsistent categories

```
df['Health_Impacts'].fillna('None', inplace=True)
```

• It was observed that the column **“Health_Impacts”** contained 6494 missing entries. These missing values likely represented cases where no specific health issue was recorded. To maintain data consistency and avoid losing rows with incomplete information, I decided to replace all missing values in this column with the label `'None'`.

```
(df['Avg_Daily_Screen_Time_hr'] == 0.0).sum()
```

• This line counts how many rows in the column **“Avg_Daily_Screen_Time_hr”** have a value of 0.0 hours. Such entries indicate users with no recorded screen time, which might be unrealistic or represent missing data recorded as zero.

```
df['Avg_Daily_Screen_Time_hr'] = df.groupby('Gender')['Avg_Daily_Screen_Time_hr'] \
                                   .transform(lambda x: x.fillna(x.mean()))
```

• This line handles **zero screen time values (0.0)** by replacing them with the **average screen time for each gender group**. Instead of keeping zeros, which may represent unrealistic or incorrect entries, the code computes the **mean screen time separately for males and females**.This approach ensures more realistic data representation and avoids bias that could arise from using an overall average.

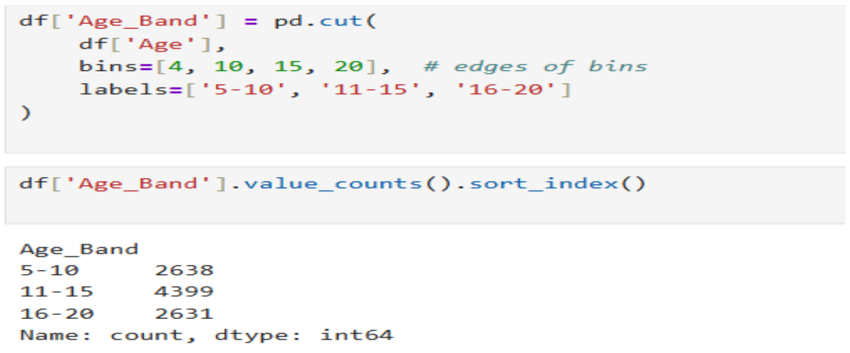
6. Handle Duplicates:

```
df.duplicated().sum()
df.drop_duplicates(inplace=True)
```

• The code identifies and removes duplicate entries from a dataset or list to ensure that each record is unique. This is important because duplicates can distort analysis, calculations, or results.

7. Create derived fields :

• **Age Bands**



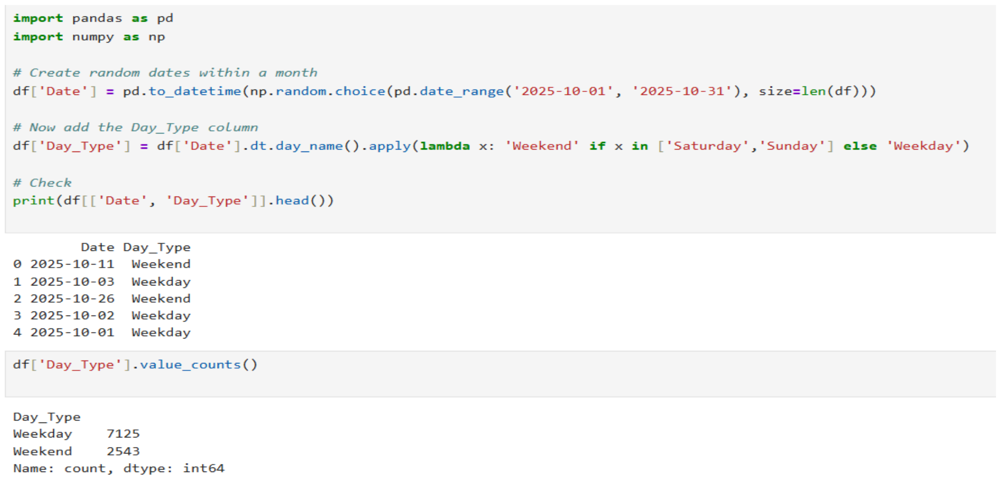
```
df['Age_Band'] = pd.cut(
    df['Age'],
    bins=[4, 10, 15, 20],  # edges of bins
    labels=['5-10', '11-15', '16-20']
)
```

```
df['Age_Band'].value_counts().sort_index()
```

```
Age_Band
5-10      2638
11-15     4399
16-20     2631
Name: count, dtype: int64
```

A new column named **'Age_Band'** was created using the **`pd.cut()`** function to categorize the age values into specific ranges. The age data was divided into three intervals — 5–10, 11–15, and 16–20 — to group individuals based on their age categories.This classification makes it easier to analyze patterns and trends across different age groups in the dataset. **df['Age_Band'].value_counts().sort_index()** was used to count the number of records in each age group and display them in ascending order of the age bands for clear interpretation.

• **Weekday/Weekend Flags**



```
import pandas as pd
import numpy as np

# Create random dates within a month
df['Date'] = pd.to_datetime(np.random.choice(pd.date_range('2025-10-01', '2025-10-31'), size=len(df)))

# Now add the Day_Type column
df['Day_Type'] = df['Date'].dt.day_name().apply(lambda x: 'Weekend' if x in ['Saturday','Sunday'] else 'Weekday')

# Check
print(df[['Date', 'Day_Type']].head())
```

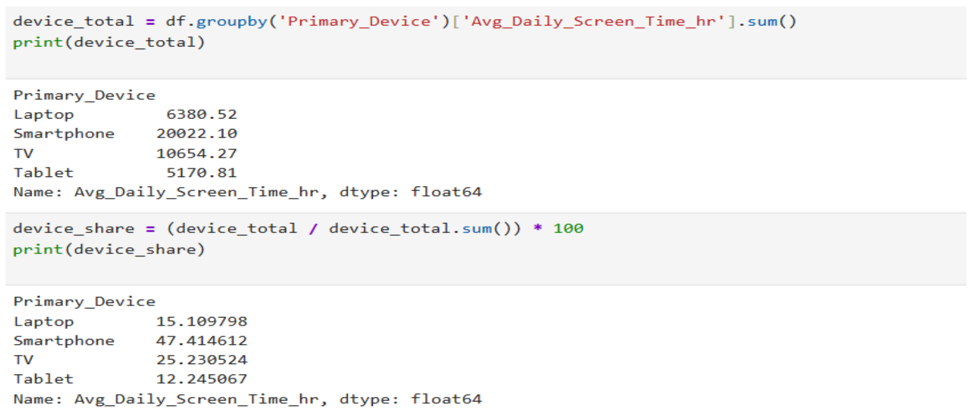
```
        Date Day_Type
0 2025-10-11  Weekend
1 2025-10-03  Weekday
2 2025-10-26  Weekend
3 2025-10-02  Weekday
4 2025-10-01  Weekday
```

```
df['Day_Type'].value_counts()
```

```
Day_Type
Weekday    7125
Weekend    2543
Name: count, dtype: int64
```

Random dates were generated for each record within **October 2025** using the **pd.date_range()** and **np.random.choice()** functions, and these were stored in a new column called **'Date'**. From this column, another column named **'Day_Type'** was created to identify whether each date falls on a **Weekday** or a **Weekend**, based on the day name extracted using **dt.day_name().** The condition classified **Saturday** and **Sunday** as **Weekend**, while all other days were labeled as **Weekday**. To verify the distribution, the command **df['Day_Type'].value_counts()** was used, which counts how many entries fall into each category. This helps in understanding the proportion of weekdays and weekends in the dataset.

• **Device/Activity shares**



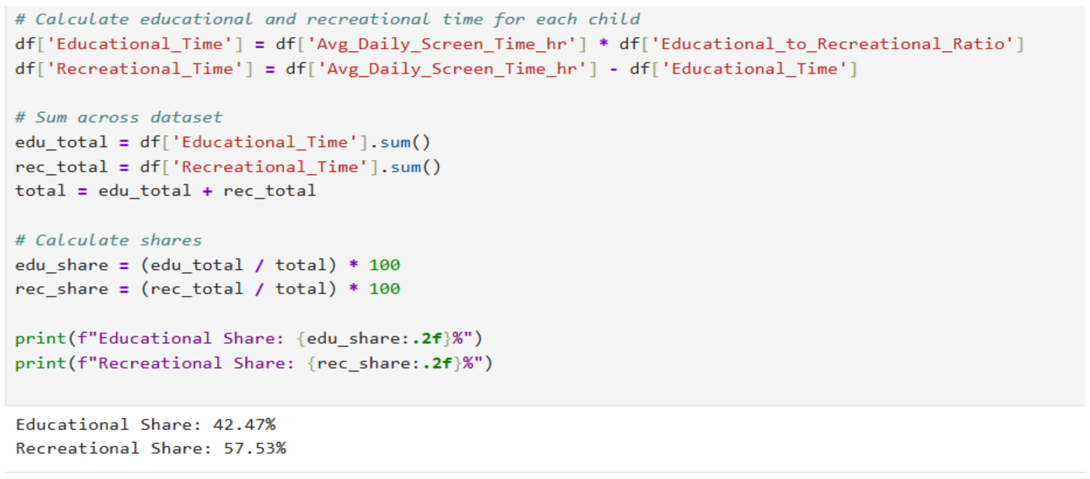
```
device_total = df.groupby('Primary_Device')['Avg_Daily_Screen_Time_hr'].sum()
print(device_total)
```

```
Primary_Device
Laptop         6380.52
Smartphone    20022.10
TV            10654.27
Tablet         5170.81
Name: Avg_Daily_Screen_Time_hr, dtype: float64
```

```
device_share = (device_total / device_total.sum()) * 100
print(device_share)
```

```
Primary_Device
Laptop        15.109798
Smartphone    47.414612
TV            25.230524
Tablet        12.245067
Name: Avg_Daily_Screen_Time_hr, dtype: float64
```

The dataset was grouped by **'Primary_Device'** to calculate the total average daily screen time for each device using **df.groupby('Primary_Device')['Avg_Daily_Screen_Time_hr'].sum().** This result, stored in **device_total**, represents total screen usage per device type. Then, **(device_total / device_total.sum()) * 100** was used to find each device's percentage share, stored as **device_share**. This analysis helps identify which device contributes most to overall screen time .

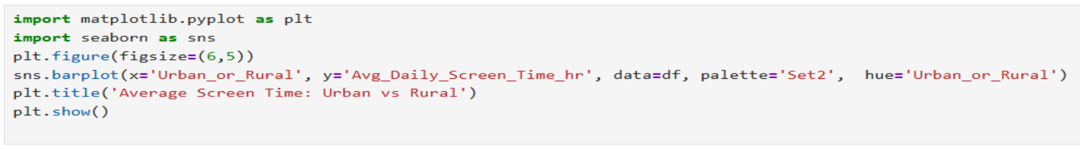


```
# Calculate educational and recreational time for each child
df['Educational_Time'] = df['Avg_Daily_Screen_Time_hr'] * df['Educational_to_Recreational_Ratio']
df['Recreational_Time'] = df['Avg_Daily_Screen_Time_hr'] - df['Educational_Time']

# Sum across dataset
edu_total = df['Educational_Time'].sum()
rec_total = df['Recreational_Time'].sum()
total = edu_total + rec_total

# Calculate shares
edu_share = (edu_total / total) * 100
rec_share = (rec_total / total) * 100

print(f"Educational Share: {edu_share:.2f}%")
print(f"Recreational Share: {rec_share:.2f}%")
```

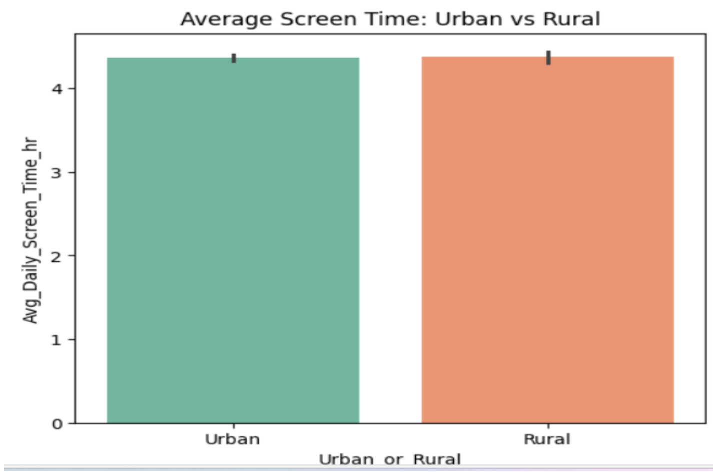
```
Educational Share: 42.47%
Recreational Share: 57.53%
```

For each child, the **educational** and **recreational** screen times were calculated based on the average daily screen time and the **Educational_to_Recreational_Ratio**. The educational time was obtained by multiplying the average screen time with this ratio, while the recreational time was found by subtracting it from the total screen time. The total educational and recreational times were then summed across the dataset to get overall values. Finally, the percentage share of each category was calculated to determine how much screen time was spent on educational versus recreational activities.

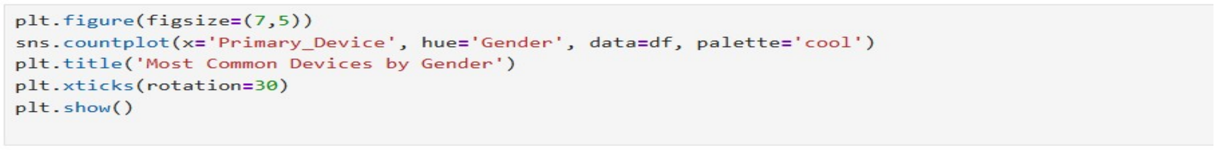
8.Data Visualization:



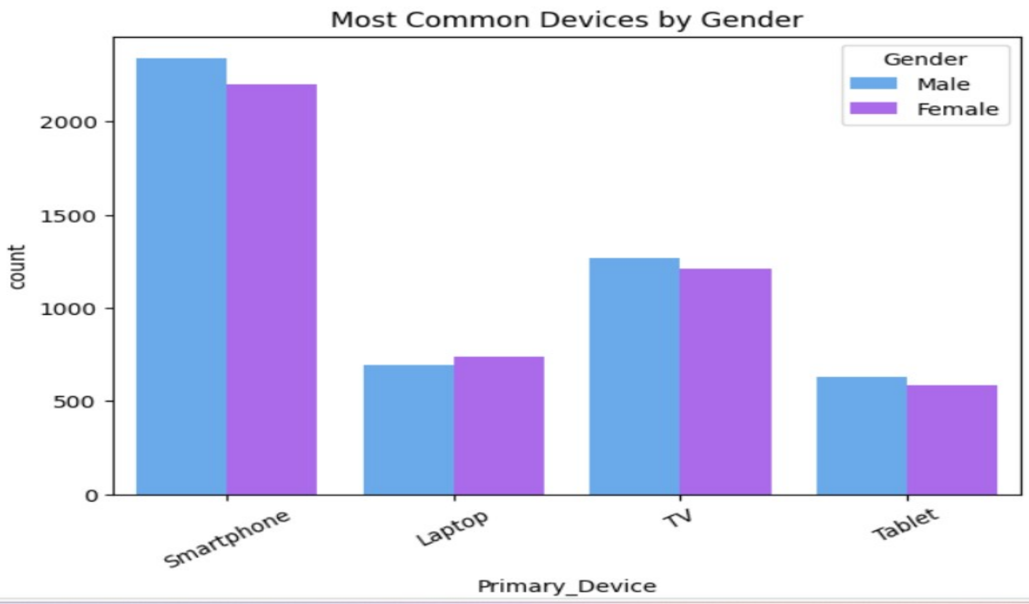
```
import matplotlib.pyplot as plt
import seaborn as sns
plt.figure(figsize=(6,5))
sns.barplot(x='Urban_or_Rural', y='Avg_Daily_Screen_Time_hr', data=df, palette='Set2',  hue='Urban_or_Rural')
plt.title('Average Screen Time: Urban vs Rural')
plt.show()
```



• This code creates a bar chart to compare the **average daily screen time** between children living in **urban** and **rural** areas.

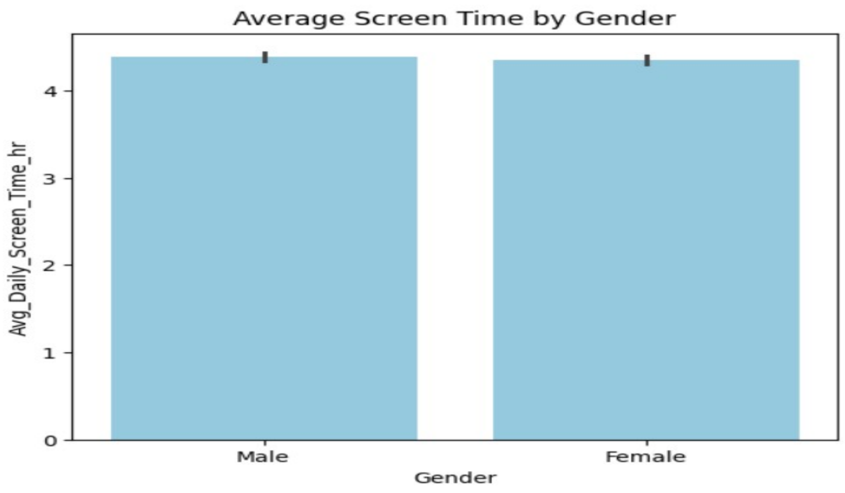


```
plt.figure(figsize=(7,5))
sns.countplot(x='Primary_Device', hue='Gender', data=df, palette='cool')
plt.title('Most Common Devices by Gender')
plt.xticks(rotation=30)
plt.show()
```



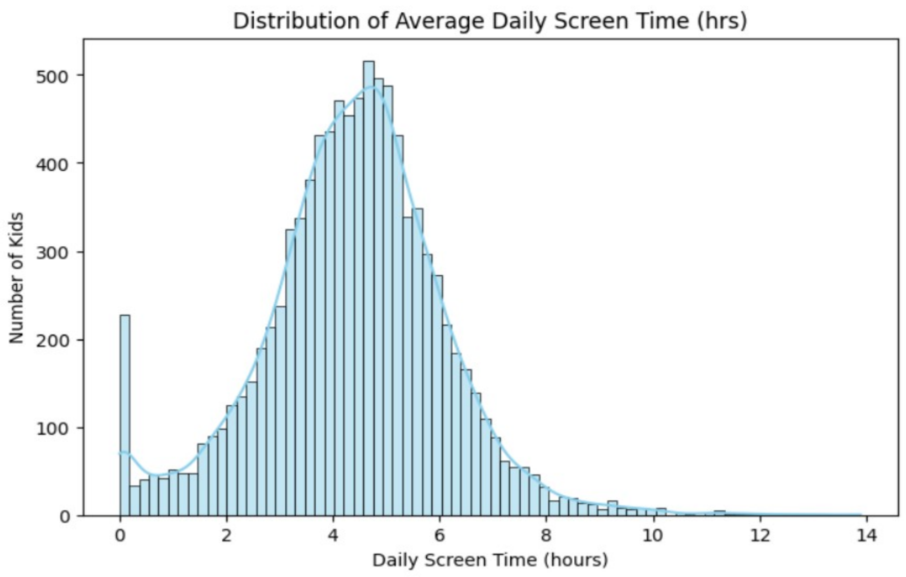
• This code creates a **count plot** to show which devices are most commonly used by children, separated by gender.It helps visualize the **distribution of device usage** among boys and girls.

```
plt.figure(figsize=(6,5))
sns.barplot(x='Gender', y='Avg_Daily_Screen_Time_hr', data=df, color='skyblue')
plt.title('Average Screen Time by Gender')
plt.show()
```



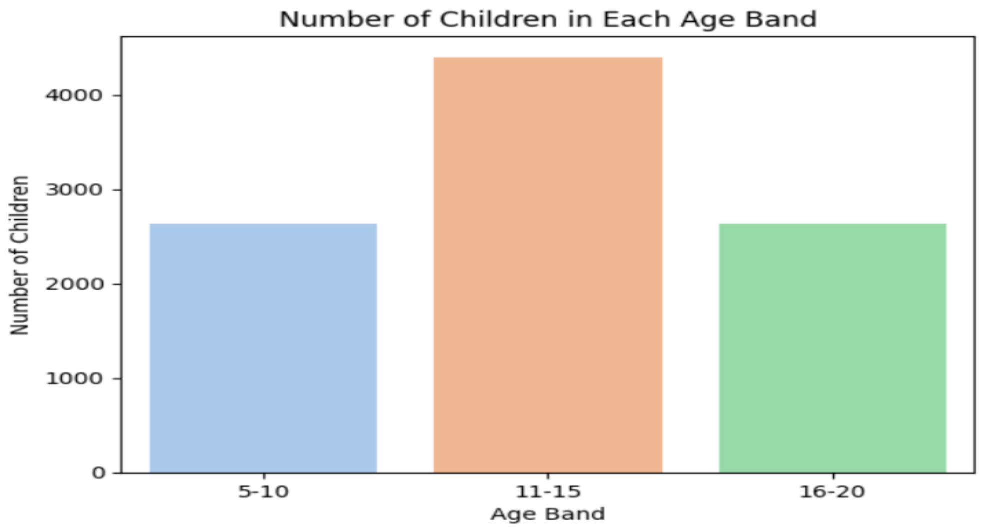
• This code creates a **bar chart** to compare the **average daily screen time** between boys and girls.It helps visualize whether gender influences screen time habits.

```
plt.figure(figsize=(8,5))
sns.histplot(df['Avg_Daily_Screen_Time_hr'], kde=True, color='skyblue')
plt.title('Distribution of Average Daily Screen Time (hrs)')
plt.xlabel('Daily Screen Time (hours)')
plt.ylabel('Number of Kids')
plt.show()
```



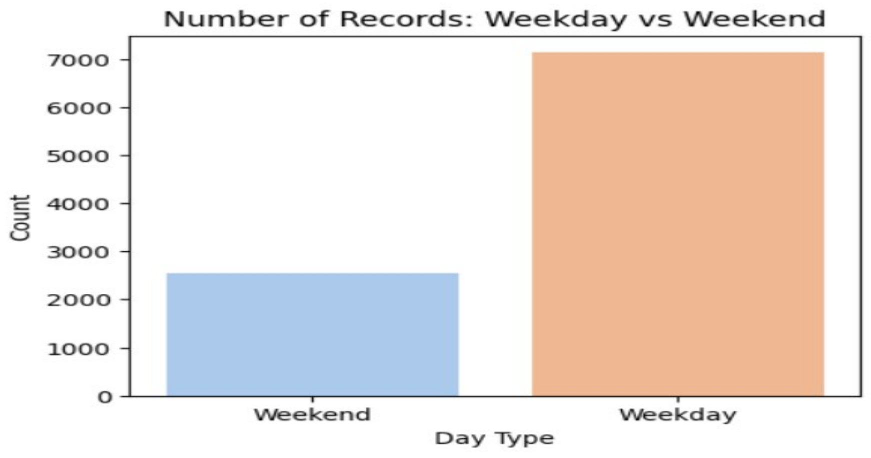
• This code creates a **count plot** to show the number of children in each **Age_Band**. It helps visualize how children are distributed across different age groups and quickly compare the size of each group.

```
sns.countplot(x='Age_Band',hue='Age_Band',data=df, palette='pastel',legend='auto')

plt.title('Number of Children in Each Age Band')
plt.xlabel('Age Band')
plt.ylabel('Number of Children')
plt.show()
```

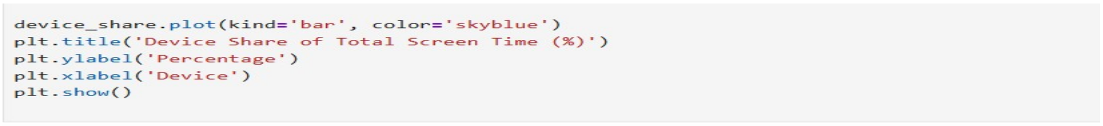


- This code creates a **count plot** to show the number of children in each **Age_Band**. It helps visualize how children are distributed across different age groups and quickly compare the size of each group.

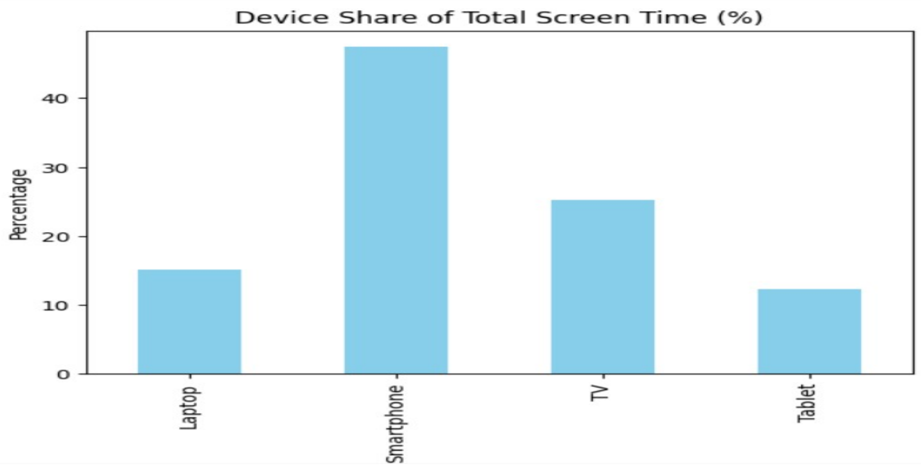
```
plt.figure(figsize=(5,4))
sns.countplot(x='Day_Type',hue='Day_Type', data=df, palette='pastel',legend=False)
plt.title('Number of Records: Weekday vs Weekend')
plt.xlabel('Day Type')
plt.ylabel('Count')
plt.show()
```



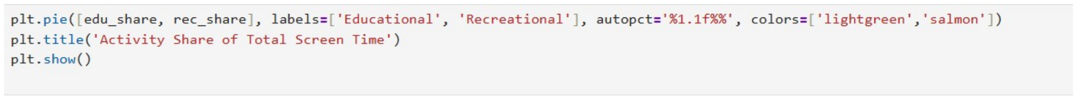
• This code creates a **count plot** to compare the number of records for **Weekdays** and **Weekends**. It helps visualize how the data is distributed across day types and quickly shows whether there are more weekday or weekend entries.



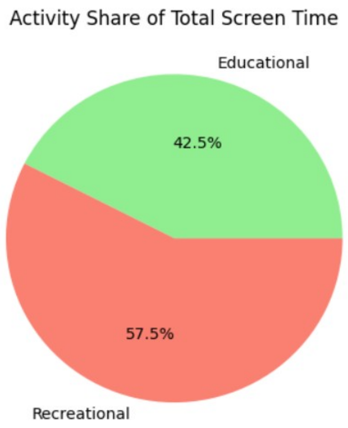
```
device_share.plot(kind='bar', color='skyblue')
plt.title('Device Share of Total Screen Time (%)')
plt.ylabel('Percentage')
plt.xlabel('Device')
plt.show()
```



• This code creates a **bar chart** to display the percentage share of total screen time for each **Primary_Device**. It helps visually compare which devices contribute most to overall screen usage.



```
plt.pie([edu_share, rec_share], labels=['Educational', 'Recreational'], autopct='%1.1f%%', colors=['lightgreen','salmon'])
plt.title('Activity Share of Total Screen Time')
plt.show()
```



• This code creates a **pie chart** to show the proportion of **Educational** and **Recreational** screen time.It helps easily visualize how children’s total screen time is divided between learning and leisure activities.

9. Save the processed dataset to a CSV file named **‘preprocessed_data.csv’** without including the index. It ensures that all the newly created columns, such as **Age_Band**, **Day_Type**, and activity times, are stored for future use.