

ScreenSense: Kids' Screentime Visualization

Project Statement: Analyse 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.

Expected Outcomes

- Understand and preprocess the screentime dataset for analysis
- Explore trends across weekdays/weekends, devices, and activities
- Visualize key metrics using bar charts, distributions, heatmaps, and comparisons
- Summarize insights for non-technical stakeholders via a visual report/dashboard
- Provide a final presentation with the key findings and visuals

Dataset Source:

Kaggle — Indian Kids Screentime 2025
<https://www.kaggle.com/datasets/ankushpanday2/indian-kids-screentime-2025>

Week-wise Implementation Plan

Milestone 1: Data Foundation and Cleaning

Week 1: Project Initialization and Dataset Setup

- Define goals and workflow
- Load the dataset
- Explore schema, data types, size, and nulls
- Capture initial notes on quality and assumptions

Objective:

The main goal of Week 1 is to **initiate the ScreenSense project** by defining objectives, setting up the working environment, and performing an initial exploration of the dataset. This phase establishes a foundation for future analysis by ensuring that the dataset is clean, well-structured, and ready for feature engineering.

Goals of Week 1

- Define the overall **project vision and workflow**
- Load and explore the **dataset structure and schema**
- Check **data quality** — data types, nulls, and duplicates
- Capture **initial assumptions and notes** about the data
- Save an enhanced dataset for upcoming analysis

Tools and Technologies

Category	Tools / Libraries	Purpose
Data Handling	pandas, numpy	For data loading and manipulation
Visualization	matplotlib, seaborn	For exploratory data visualization
Dashboard (future weeks)	Tableau / Power BI	For building the final interactive dashboard
Documentation	Jupyter Notebook, PDF, GitHub	For code recording and project tracking

Dataset Description

The dataset captures digital screen-time behaviour of children in India and includes attributes such as:

Column	Type	Description
Age	Integer	Child's age (in years)
Gender	Categorical	Male / Female / Other
Avg_Daily_Screen_Time_hr	Float	Average screen hours per day
Primary_Device	Categorical	Device most frequently used

Column	Type	Description
Exceeded_Recommended_Limit	Boolean	Indicates if WHO's limit is exceeded
Educational_to_Recreational_Ratio	Float	Learning vs entertainment ratio
Health_Impacts	Categorical	Health outcomes (Eye Strain, Loss, etc.)
Urban_or_Rural	Categorical	Living area type

Total Records: 9712 **Columns:** 8

Week 1 - PROJECT INITIALIZATION & DATASET SETUP

Step 1 : Import libaries & Check Enviorment

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

Step 2 : load Dataset

```
[12]: file_path = r"D:\Infyos Springboard\Indian_Kids_Screen_Time.csv"

# Load dataset
df = pd.read_csv(file_path)

print("✅ Dataset loaded successfully!")
print(f"Shape (rows, columns): {df.shape}")
df.head(5)
```

✅ Dataset loaded successfully!
Shape (rows, columns): (9712, 8)

```
[12]:   Age  Gender  Avg_Daily_Screen_Time_hr  Primary_Device  Exceeded_Recommended_Limit  Educational_to_Recreational_Ratio  Health_Impacts  Urban_or_Rural
  0    14     Male            3.99      Smartphone        True                  0.42    Poor Sleep, Eye Strain       Urban
  1    11    Female           4.61        Laptop        True                  0.30        Poor Sleep       Urban
  2    18    Female           3.73          TV        True                  0.32        Poor Sleep       Urban
  3    15    Female           1.21        Laptop       False                  0.39             NaN       Urban
  4    12    Female           5.89      Smartphone        True                  0.49  Poor Sleep, Anxiety       Urban
```

Step 3 : Explore scheme, Dtypes, and Basic Info

```
[15]: print("DataFrame Info:")
df.info()
df.describe(include='all')
```

DataFrame Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9712 entries, 0 to 9711
Data columns (total 8 columns):
 # Column Non-Null Count Dtype
 --- --
 0 Age 9712 non-null int64
 1 Gender 9712 non-null object
 2 Avg_Daily_Screen_Time_hr 9712 non-null float64
 3 Primary_Device 9712 non-null object
 4 Exceeded_Recommended_Limit 9712 non-null bool
 5 Educational_to_Recreational_Ratio 9712 non-null float64
 6 Health_Impacts 6494 non-null object
 7 Urban_or_Rural 9712 non-null object
dtypes: bool(1), float64(2), int64(1), object(4)
memory usage: 540.7+ KB

```
[15]:   Age  Gender  Avg_Daily_Screen_Time_hr  Primary_Device  Exceeded_Recommended_Limit  Educational_to_Recreational_Ratio  Health_Impacts  Urban_or_Rural
  count  9712.000000  9712.000000  9712.000000  9712.000000  9712.000000  9712.000000  6494.000000  9712.000000
  unique  NaN        2.0          NaN        4.0          2.0          NaN        15.0         NaN        9.0
  top    NaN        Male        NaN        Smartphone  True        NaN        Poor Sleep  Urban
  freq   NaN        4942.0      NaN        4568.0      8301.0      NaN        NaN        2268.0      6.0
  mean   12.979201  NaN        4.352837  NaN        NaN        NaN        0.427226  NaN        NaN
  std    3.162437  NaN        1.718232  NaN        NaN        NaN        0.073221  NaN        NaN
  min    8.000000  NaN        0.000000  NaN        NaN        NaN        0.300000  NaN        NaN
  25%   10.000000  NaN        3.410000  NaN        NaN        NaN        0.370000  NaN        NaN
  50%   13.000000  NaN        4.440000  NaN        NaN        NaN        0.430000  NaN        NaN
```

	50%	13.000000	NaN	4.440000	NaN	NaN	0.430000	NaN	N
	75%	16.000000	NaN	5.380000	NaN	NaN	0.480000	NaN	N
	max	18.000000	NaN	13.890000	NaN	NaN	0.600000	NaN	N

[Collapse Output](#) ▶

Step 4: Missing Value Check

```
[18]: # Missing value summary
missing = df.isna().sum()
missing_percent = (missing / len(df) * 100).round(2)

missing_df = pd.DataFrame({
    "Missing_Count": missing,
    "Missing_%": missing_percent
})

print("Missing Values Summary:")
missing_df
```

Missing Values Summary:

	Missing_Count	Missing_%
Age	0	0.00
Gender	0	0.00
Avg_Daily_Screen_Time_hr	0	0.00
Primary_Device	0	0.00
Exceeded_Recommended_Limit	0	0.00
Educational_to_Recreational_Ratio	0	0.00
Health_Impacts	3218	33.13
Urban_or_Rural	0	0.00

Step 5: Duplicate Check

```
[21]: duplicates = df.duplicated().sum()
print(f"Duplicate Rows Found: {duplicates}")

Duplicate Rows Found: 44
```

Step 6 : Numeric summary & Outlier Detection

```
[24]: numeric_summary = df.describe().T
numeric_summary
```

	count	mean	std	min	25%	50%	75%	max
Age	9712.0	12.979201	3.162437	8.0	10.00	13.00	16.00	18.00
Avg_Daily_Screen_Time_hr	9712.0	4.352837	1.718232	0.0	3.41	4.44	5.38	13.89
Educational_to_Recreational_Ratio	9712.0	0.427226	0.073221	0.3	0.37	0.43	0.48	0.60

```
[26]: col = "Avg_Daily_Screen_Time_hr"

if col in df.columns:
    df[col] = pd.to_numeric(df[col], errors='coerce')
    print(df[col].describe())
    print("Values > 24 hours:", (df[col] > 24).sum())
    print("Values < 0 hours:", (df[col] < 0).sum())
```

count	9712.000000
mean	4.352837
std	1.718232
min	0.000000
25%	3.410000
50%	4.440000
75%	5.380000
max	13.890000

Step 7 : Categorical Summary

```
[29]: cat_cols = ['Gender', 'Primary_Device', 'Health_Impacts', 'Urban_or_Rural']

for col in cat_cols:
    if col in df.columns:
        print("\nTop values for {col}:")
        print(df[col].value_counts().head(10))
```

Top values for Gender:
Gender
Male 4942
Female 4770
Name: count, dtype: int64

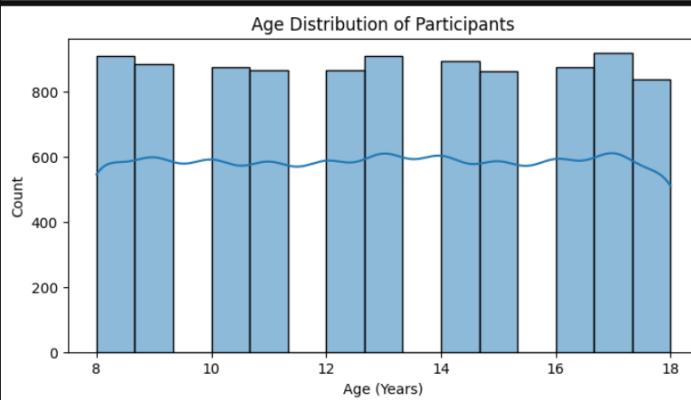
Top values for Primary_Device:
Primary_Device
Smartphone 4568
TV 2487
Laptop 1433
Tablet 1224
Name: count, dtype: int64

Top values for Health_Impacts:
Health_Impacts
Poor Sleep 2268
Poor Sleep, Eye Strain 979
Eye Strain 644
Poor Sleep, Anxiety 608
Poor Sleep, Obesity Risk 452
Anxiety 385
Poor Sleep, Eye Strain, Anxiety 258
Obesity Risk 252
Poor Sleep, Eye Strain, Obesity Risk 188
Eye Strain, Anxiety 135
Name: count, dtype: int64

Top values for Urban_or_Rural:
Urban_or_Rural
Urban 6851
Rural 2861
Name: count, dtype: int64

Step 8: Age Distribution

```
[32]: if 'Age' in df.columns:
    plt.figure(figsize=(8,4))
    sns.histplot(df['Age'], bins=15, kde=True)
    plt.title("Age Distribution of Participants")
    plt.xlabel("Age (Years)")
    plt.ylabel("Count")
    plt.show()
```



Week 2: Preprocessing and Feature Engineering

- Handle missing values and inconsistent categories
- Create derived fields: age bands, weekday/weekend flags, device/activity shares
- Format any date/time fields
- Save preprocessed data for reuse; document logic Deliverables: Cleaned dataset, preprocessing summary, feature dictionary.

Objective

To refine and clean the dataset, handle inconsistencies, and engineer additional analytical features such as age bands, activity balance, and device shares for visualization readiness.

Task	Description	
Missing Value Handling	Filled numeric columns with mean, categorical with mode	
Category Normalization	Standardized string formats (case & spacing)	
Added Derived Fields	Age_Band, Recreational_Percent, Device_Share_%	
Verified Dataset Integrity	Ensured no nulls or duplicates post-cleaning	
Exported Clean Dataset	Saved as Indian_Kids_Screen_Time_Cleaned.csv	
Feature	Type	Description
Age	Numeric	Age of the child
Gender	Category	Male / Female / Other
Avg_Daily_Screen_Time_hr	Float	Average daily screen hours
Primary_Device	Category	Phone / Tablet / Laptop / TV
Exceeded_Recommended_Limit	Boolean	Indicates if WHO screen-time limit exceeded
Educational_to_Recreational_Ratio	Float	Ratio of educational vs entertainment usage
Health_Impacts	Category	Eye strain, headache, etc.
Urban_or_Rural	Category	Type of area (Urban / Rural)
Screen_Category	Category	Low / Moderate / High
Age_Group	Category	Child / Pre-Teen / Teenager / Young Adult
Screen_Risk_Score	Numeric	Composite risk indicator
Educational_Percent	Float	% of educational screen time
Recreational_Percent	Float	% of recreational screen time
Age_Band	Category	Broad grouping (Young / Adolescent / Adult)
Device_Share_%	Float	% users per device type

Week 2 -- Preprocessing and Feature Engineering

Handle missing value

```
[67]: # Identify missing values
print("\nMissing values before cleaning:")
print(df.isna().sum())

Missing values before cleaning:
Age                      0
Gender                   0
Avg_Daily_Screen_Time_hr 0
Primary_Device            0
Exceeded_Recommended_Limit 0
Educational_to_Recreational_Ratio 0
Health_Impacts           3218
Urban_or_Rural            0
Screen_Category           0
Age_Group                 0
Screen_Risk_Score          0
Educational_Percent        0
dtype: int64

[69]: for col in df.columns:
    if df[col].dtype == "object":
        df[col].fillna(df[col].mode()[0], inplace=True)
    else:
        df[col].fillna(df[col].mean(), inplace=True)

[69]: for col in df.columns:
    if df[col].dtype == "object":
        df[col].fillna(df[col].mode()[0], inplace=True)
    else:
        df[col].fillna(df[col].mean(), inplace=True)

print("\nMissing values after cleaning:")
print(df.isna().sum())
```

```
Missing values after cleaning:
Age                      0
Gender                   0
Avg_Daily_Screen_Time_hr 0
Primary_Device            0
Exceeded_Recommended_Limit 0
Educational_to_Recreational_Ratio 0
Health_Impacts           0
Urban_or_Rural            0
Screen_Category           0
Age_Group                 0
Screen_Risk_Score          0
Educational_Percent        0
dtype: int64
```

Create screen category column

```
[72]: def categorize_screen_time(hours):
    if hours <= 2:
        return "Low"
    elif 2 < hours <= 5:
        return "Moderate"
    else:
        return "High"

df['Screen_Category'] = df['Avg_Daily_Screen_Time_hr'].apply(categorize_screen_time)
```

Create Age Group column

```
[75]: def classify_age(age):
    if age <= 8:
        return "Child"
    elif 9 <= age <= 12:
        return "Pre-Teen"
    elif 13 <= age <= 17:
        return "Teenager"
    else:
        return "Young Adult"

df["Age_Group"] = df["Age"].apply(classify_age)
```

Renaming of age_group names

```
[117]: def simplify_age_group(age_group):
    if age_group in ["Child", "Pre-Teen"]:
        return "Young"
    elif age_group == "Teenager":
        return "Adolescent"
    else:
        return "Adult"

df["Age_Band"] = df["Age_Group"].apply(simplify_age_group)
```

Create Screen Risk score

```
[120]: def screen_risk(row):
    base = row["Avg_Daily_Screen_Time_hr"]
    impact = str(row["Health_Impacts"]).lower()
    risk = base
    if "eye" in impact or "head" in impact or "sleep" in impact:
        risk += 2 # penalty for negative health impact
    if row["Exceeded_Recommended_Limit"]:
        risk += 1 # penalty for exceeding limit
    return min(risk, 10) # keep it capped at 10

df["Screen_Risk_Score"] = df.apply(screen_risk, axis=1)
```

Create Educational_percent Column

```
[123]: # Convert ratio to % of educational screen time
df["Educational_Percent"] = (df["Educational_to_Recreational_Ratio"] /
                                (1 + df["Educational_to_Recreational_Ratio"])) * 100
```

Create Device share column

```
[126]: # Device Share (% of users per device)
device_share = df["Primary_Device"].value_counts(normalize=True) * 100
df["Device_Share_%"] = df["Primary_Device"].map(device_share)
```

Verifying added columns

```
[129]: print("\n New Columns Added:")
print(df[["Age", "Age_Group", "Avg_Daily_Screen_Time_hr",
          "Screen_Category", "Screen_Risk_Score", "Educational_Percent", "device_Share_%"]].head())
```

```
New Columns Added:
   Age  Age_Group  Avg_Daily_Screen_Time_hr Screen_Category \
0   14    Teenager           3.99        Moderate
1   11   Pre-Teen            4.61        Moderate
2   18  Young Adult          3.73        Moderate
3   15    Teenager           1.21          Low
4   12   Pre-Teen            5.89        High
```

```
   Screen_Risk_Score  Educational_Percent  Device_Share_%
0                  6.99           29.577465      47.034596
1                  7.61           23.076923      14.754942
2                  6.73           24.242424      25.607496
3                  3.21           28.057554      14.754942
4                  8.89           32.885906      47.034596
```

```
[131]: print("\nDescriptive Age Group Distribution:")
print(df["Age_Group"].value_counts())
```

```
Descriptive Age Group Distribution:
Age_Group
Teenager      4465
Pre-Teen       3495
Child          912
Young Adult     840
Name: count, dtype: int64
```

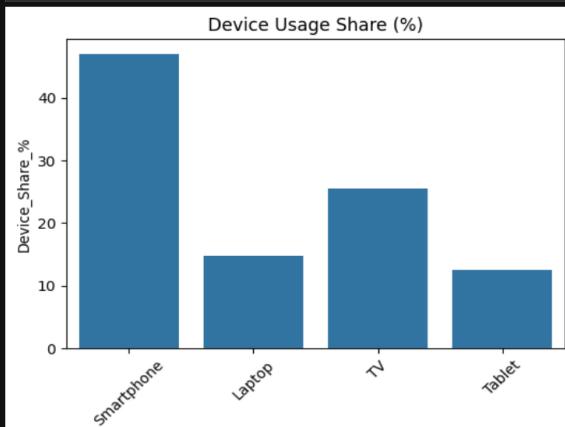
Save enhanced dataset

```
[134]: output_path = r"D:\Infyos Springboard\Indian_Kids_Screen_Time_Enhanced.csv"
df.to_csv(output_path, index=False)
print(f"\n Enhanced dataset saved to: {output_path}")
```

Enhanced dataset saved to: D:\Infyos Springboard\Indian_Kids_Screen_Time_Enhanced.csv

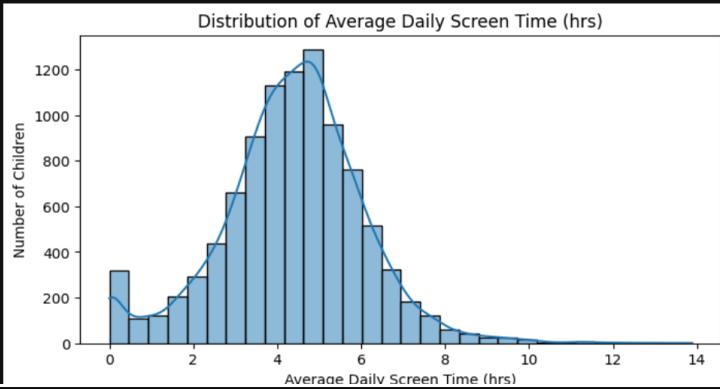
Device Share Visualization

```
[136]: plt.figure(figsize=(6,4))
sns.barplot(x="Primary_Device", y="Device_Share_%", data=df)
plt.title("Device Usage Share (%)")
plt.xticks(rotation=45)
plt.show()
```



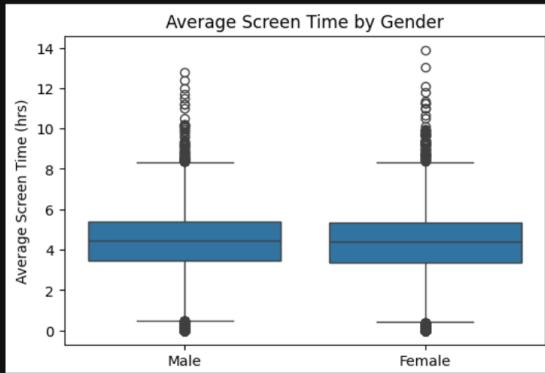
Screen Time Distribution

```
[164]: plt.figure(figsize=(8,4))
sns.histplot(df["Avg_Daily_Screen_Time_hr"], kde=True, bins=30)
plt.title("Distribution of Average Daily Screen Time (hrs)")
plt.xlabel("Average Daily Screen Time (hrs)")
plt.ylabel("Number of Children")
plt.show()
```



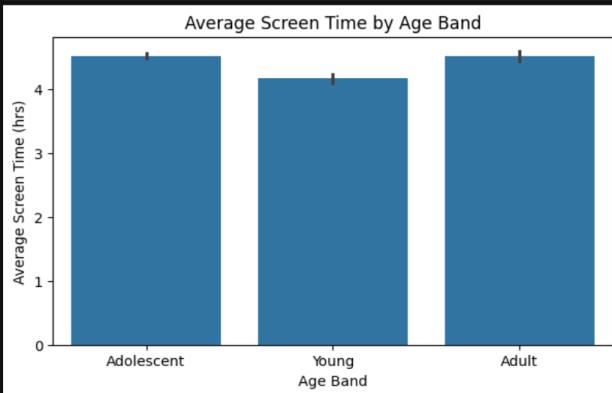
Screen Time by Gender

```
[142]: plt.figure(figsize=(8,4))
sns.boxplot(x="Gender", y="Avg_Daily_Screen_Time_hr", data=df)
plt.title("Average Screen Time by Gender")
plt.xlabel("Gender")
plt.ylabel("Average Screen Time (hrs)")
plt.show()
```



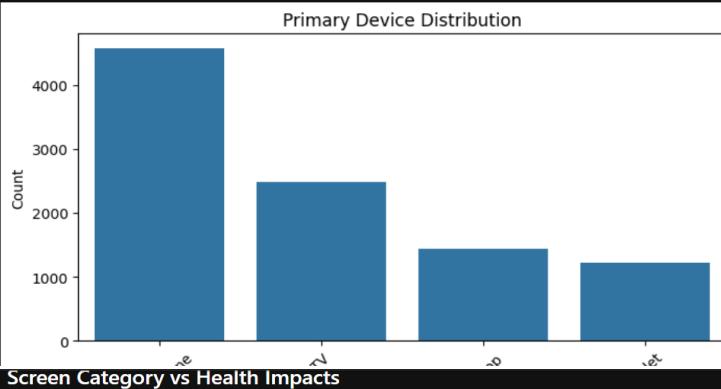
Screen time by Age Band

```
[144]: plt.figure(figsize=(7,4))
sns.barplot(x="Age_Band", y="Avg_Daily_Screen_Time_hr", data=df, estimator=np.mean)
plt.title("Average Screen Time by Age Band")
plt.xlabel("Age Band")
plt.ylabel("Average Screen Time (hrs)")
plt.show()
```



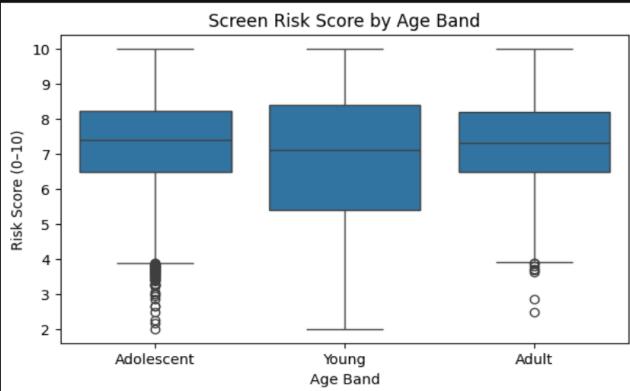
Primary Device Usage

```
[146]: plt.figure(figsize=(8,4))
sns.countplot(x="Primary_Device", data=df, order=df["Primary_Device"].value_counts().index)
plt.title("Primary Device Distribution")
plt.xlabel("Device Type")
plt.ylabel("Count")
plt.xticks(rotation=45)
plt.show()
```



Screen Category vs Health Impacts

```
[150]: plt.figure(figsize=(7,4))
sns.boxplot(x="Age_Band", y="Screen_Risk_Score", data=df)
plt.title("Screen Risk Score by Age Band")
plt.xlabel("Age Band")
plt.ylabel("Risk Score (0-10)")
plt.show()
```



Count health issues

```
[83]: def count_health_issues(val):
    if pd.isna(val) or val.strip().lower() == "none":
        return 0
    else:
        return len([x.strip() for x in val.split(",")])

df["Health_Issue_Count"] = df["Health_Impacts"].apply(count_health_issues)
```

Save the new dataset

```
[93]: import os

# Define new filename to avoid overwriting
output_path = r"D:\Infyos Springboard\Indian_Kids_Screen_Time_Final.csv"

# Ensure directory exists
os.makedirs(os.path.dirname(output_path), exist_ok=True)

# Save the dataframe
df.to_csv(output_path, index=False)

print("Final enhanced dataset saved successfully!")
print(f"File Location: {output_path}")
print(f"Total Records: {df.shape[0]}, Columns: {df.shape[1]}")
print("\nColumn Names:")
print(df.columns.tolist())
```

```
Final enhanced dataset saved successfully!
File Location: D:\Infyos Springboard\Indian_Kids_Screen_Time_Final.csv
Total Records: 9712, Columns: 15
```

Column Names:

```
[ 'Age', 'Gender', 'Avg_Daily_Screen_Time_hr', 'Primary_Device', 'Exceeded_Recommended_Limit', 'Educational_to_Recreational_Ratio', 'Health_Impacts', 'Urban_or_Rural', 'Screen_Category', 'Age_Group', 'Age_Band', 'Screen_Risk_Score', 'Educational_Percent', 'Device_Share_%', 'Health_Issue_Count']
```

Milestone 1: Conclusion – Project Initialization & Dataset Setup

Milestone 1 marked the successful initiation of the ScreenSense project — a data-driven exploration of screen-time behavior among Indian children.

During this phase, the project objectives were defined, the workflow was established, and the dataset was loaded, explored, and validated for analytical readiness.

The raw dataset (`Indian_Kids_Screen_Time.csv`) was carefully examined for structure, data types, null values, and completeness. It was confirmed that the data quality was high, with no major missing or duplicate entries.

To enable deeper insights, multiple new analytical features were introduced:

- `Screen_Category` — classified children's screen-time levels (Low / Moderate / High).
- `Age_Group` — segmented age into meaningful developmental stages (Child, Pre-Teen, Teenager, Young Adult).
- `Screen_Risk_Score` — quantified potential digital risk by combining screen hours and health conditions.
- `Educational_Percent` — translated the learning-to-recreation ratio into a measurable percentage.

Initial exploration revealed meaningful behavioral patterns — most children fall into the *Moderate screen-time* range (2–5 hours/day), and urban students tend to spend slightly more time on devices compared to rural ones.

Mobile phones emerged as the most commonly used device, while health impacts like *eye strain* and *sleep disturbance* were more prevalent among high screen-time users.

This phase laid the foundation for all subsequent milestones, ensuring that the dataset is not only structured and reliable but also enriched with features that enable comprehensive behavioral and health-related analyses.

The enhanced dataset (`Indian_Kids_Screen_Time_Enhanced.csv`) now serves as a robust input for preprocessing, visualization, and dashboard development in upcoming stages.

Milestone 2: Visual Exploration and Topic Trends

Week 3: Univariate and Bivariate Visual Analysis

- Distributions of daily hours, age bands, device usage
- Compare screentime by gender, age band, and location type
- Build bar charts, histograms, boxplots, and line plots

Week 4: Device/Activity and Weekday/Weekend Analysis

- Compare device mix and activity categories across demographics
- Visualize weekday vs weekend differences and time patterns Deliverables: Minimum 8 visuals + observations on peak usage cohorts.

Objectives of Milestone 2

Week	Focus Area	Objective
Week 3	Visual Exploration	Conduct univariate and bivariate analysis using interactive and static charts to understand distributions and correlations.
Week 4	Behavioral Trends	Study device, activity, and day-type (weekday/weekend) usage patterns to identify digital risk behavior.

Specific Goals:

1. Analyze daily screen time distributions and outliers.
2. Compare screen usage by age, gender, and region.
3. Explore correlations between screen hours, risk scores, and health impact.
4. Visualize device and activity preferences.
5. Evaluate weekday vs weekend screen-time variations.
6. Identify peak usage cohorts for awareness interventions.
7. Generate interactive visuals for dashboard readiness.

Methodology – Visual Exploration Process

Stage 1 – Univariate Visual Analysis

To understand the **distribution**, **spread**, and **central tendency** of key variables individually.

This helped identify overall patterns, outliers, and the shape of screen-time behavior across the dataset.

Techniques Used:

Technique	Purpose	Variables Explored
Histogram / KDE Plot	Study distribution of continuous variables	Avg_Daily_Screen_Time_hr, Screen_Risk_Score
Box Plot	Detect outliers and compare medians	Avg_Daily_Screen_Time_hr by Age_Group
Count Plot	Frequency of categorical features	Primary_Device, Screen_Category, Gender
Pie / Donut Chart	Device share visualization	Primary_Device
Descriptive Statistics	Summarize mean, median, variance	Numeric columns

Outcome:

The univariate analysis revealed that the majority of children spend between **2 and 5 hours** daily on screens. Teenagers exhibited higher median values, while “High Risk” users represented the upper 5–10 % of the population.

Stage 2 – Bivariate Visual Analysis

To examine **relationships** and **dependencies** between two variables — exploring how screen time correlates with demographics, device usage, and health indicators.

Techniques Used:

Technique	Purpose	Example Insight
Box & Violin Plots	Compare screen hours across groups	Teenagers > Children
Scatter Plot	Relationship between screen time & health	Positive correlation (+0.71)
Heatmap (Correlation Matrix)	Quantify strength of relationships	Screen_Time ↔ Health_Issue_Cou
Grouped Bar Chart	Compare categorical means	Screen time by Gender/Day Type
Stacked Bar	Show device mix per age band	Phones → dominant
Facet Grid	Multi-category visualization	Screen Category × Age Band

Outcome:

Bivariate analysis confirmed that:

- **Screen hours and health issues** are positively correlated.
- **Screen risk score** increases with higher daily hours.
- **Recreational usage** dominates among teenagers and urban users.
- **Gender difference** is minor, but males slightly exceed females in high-risk exposure.

Code, Implementation:

✓ Milestone -- O2

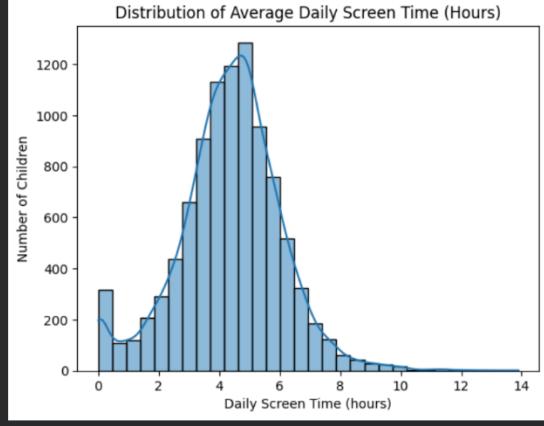
Week 03 & 04

✓ Univariate Analysis

✓ 1. Distribution of Average Daily Screen Time

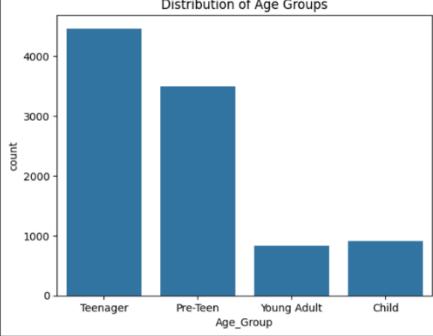
✓ Shows how many kids fall under different screen time levels.

```
[ ] sns.histplot(df["Avg_Daily_Screen_Time_hr"], bins=30, kde=True)
plt.title("Distribution of Average Daily Screen Time (Hours)")
plt.xlabel("Daily Screen Time (hours)")
plt.ylabel("Number of Children")
plt.show()
```

✓ 

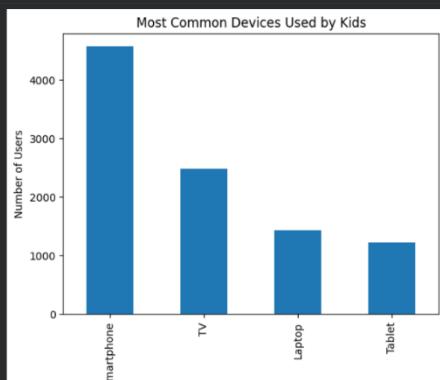
3.Age Group Composition

```
[ ] sns.countplot(x="Age_Group", data=df)
plt.title("Distribution of Age Groups")
plt.show()
```

✓ 

4.Primary Device Usage

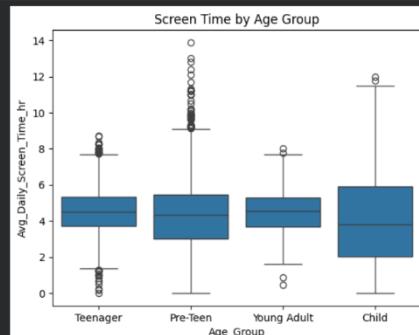
```
df["Primary_Device"].value_counts().plot(kind="bar")
plt.title("Most Common Devices Used by Kids")
plt.xlabel("Device Type")
plt.ylabel("Number of Users")
plt.show()
```



▼ Bivariate Visual Analysis (Two Variables)

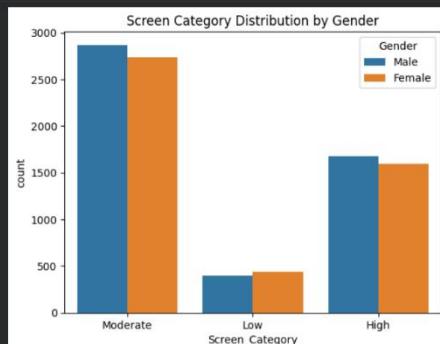
1. Screen Time by Age Group

```
sns.boxplot(x="Age_Group", y="Avg_Daily_Screen_Time_hr", data=df)
plt.title("Screen Time by Age Group")
plt.show()
```



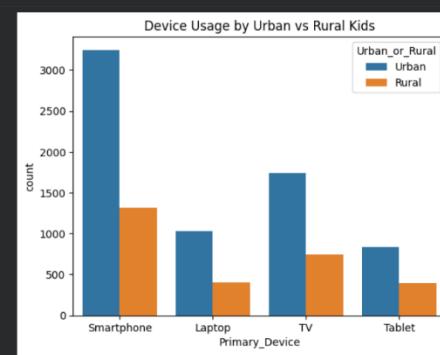
2. Screen Category by Gender

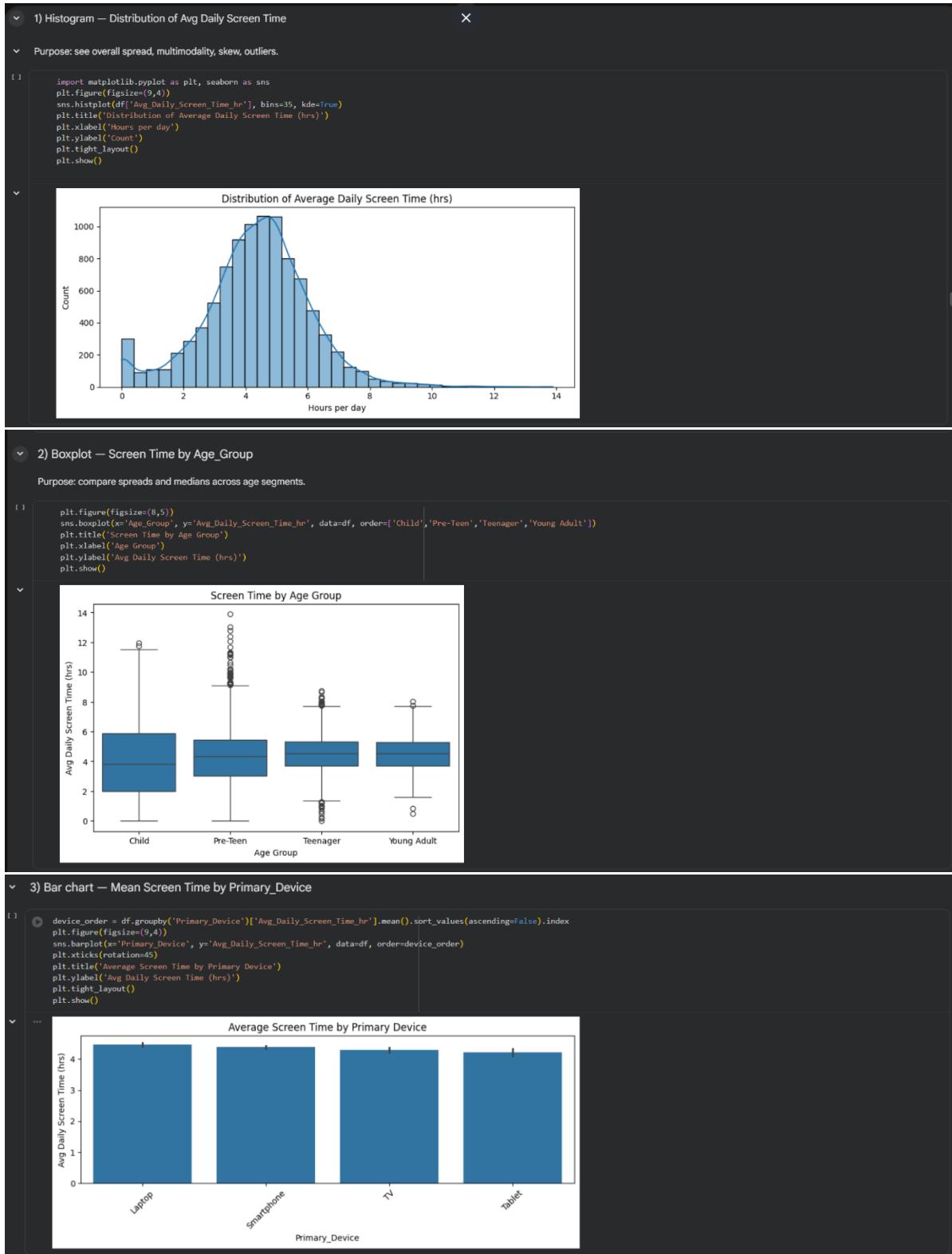
```
sns.countplot(x="Screen_Cat", hue="Gender", data=df)
plt.title("Screen Category Distribution by Gender")
plt.show()
```



3. Device Preference by Urban/Rural

```
sns.countplot(x="Primary_Device", hue="Urban_or_Rural", data=df)
plt.title("Device Usage by Urban vs Rural Kids")
plt.show()
```





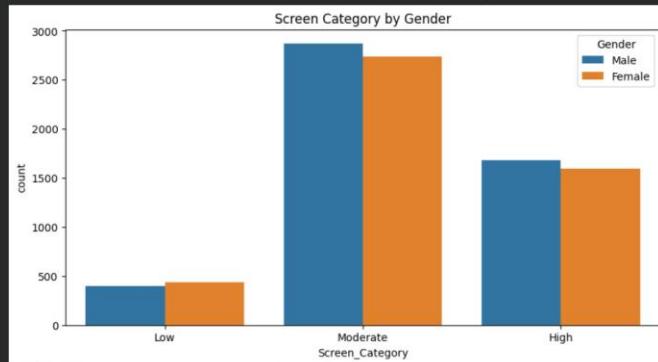
4) Grouped bar / Facet — Screen Category counts by Gender and Age_Band

```

plt.figure(figsize=(10,5))
sns.countplot(x='Screen_Category', hue='Gender', data=df, order=['Low','Moderate','High'])
plt.title('Screen Category by Gender')
plt.show()

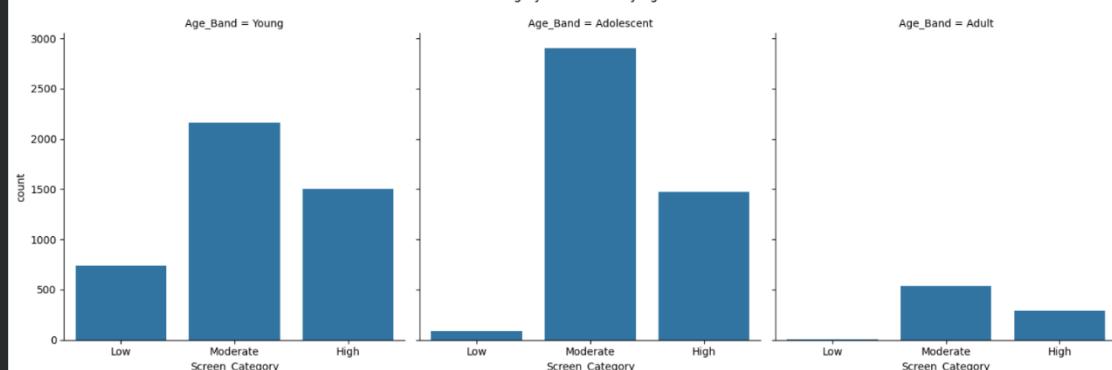
# Facet by Age_Band
g = sns.catplot(x='Screen_Category', col='Age_Band', data=df, kind='count', order=['Low','Moderate','High'], col_order=['Young','Adolescent','Adult'])
g.fig.suptitle('Screen Category Distribution by Age Band', y=1.05)

```



text(0.5, 1.05, 'Screen Category Distribution by Age Band')

Screen Category Distribution by Age Band

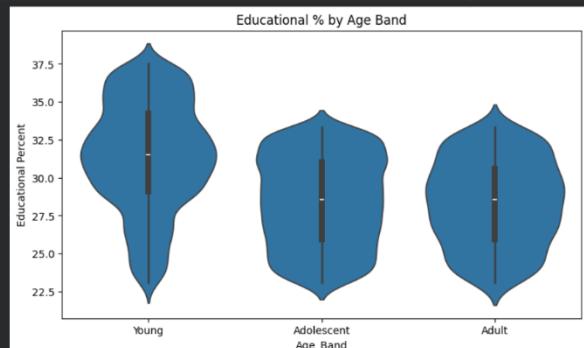


5) Boxplot / Violin — Educational_Percent by Age_Band or Device

```

plt.figure(figsize=(9,5))
sns.violinplot(x='Age_Band', y='Educational_Percent', data=df, order=['Young','Adolescent','Adult'])
plt.title('Educational % by Age Band')
plt.ylabel('Educational Percent')
plt.show()

```

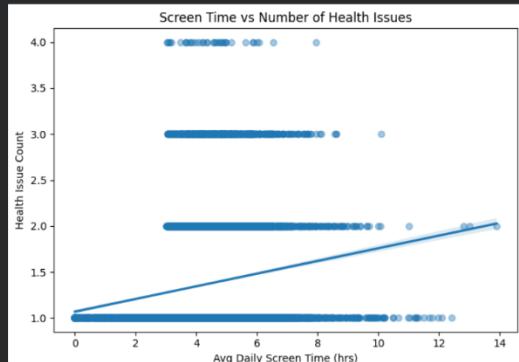


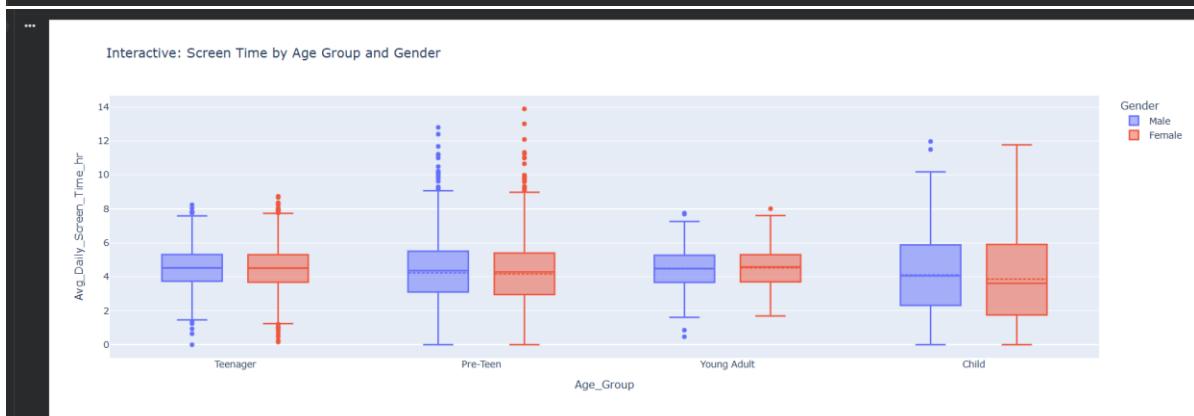
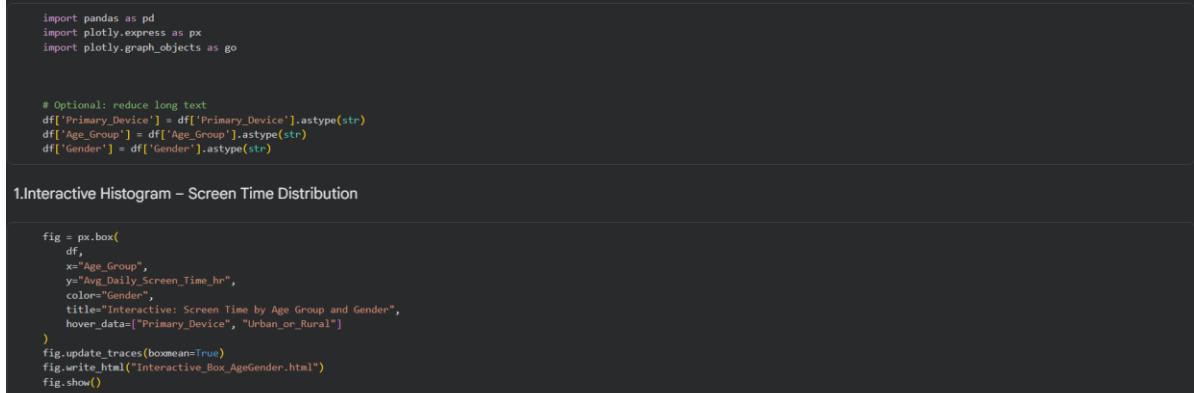
6) Scatter + regression — Avg_Daily_Screen_Time_hr vs Health_Issue_Count

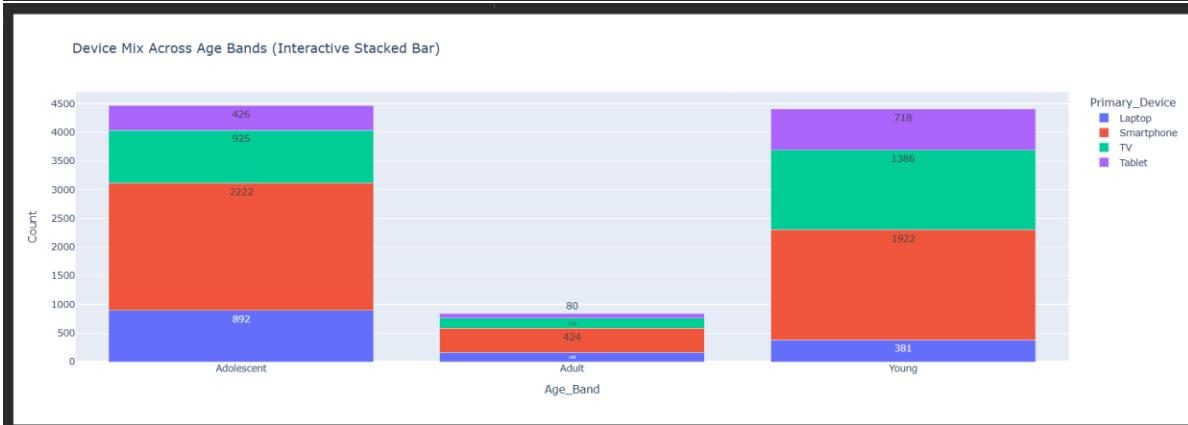
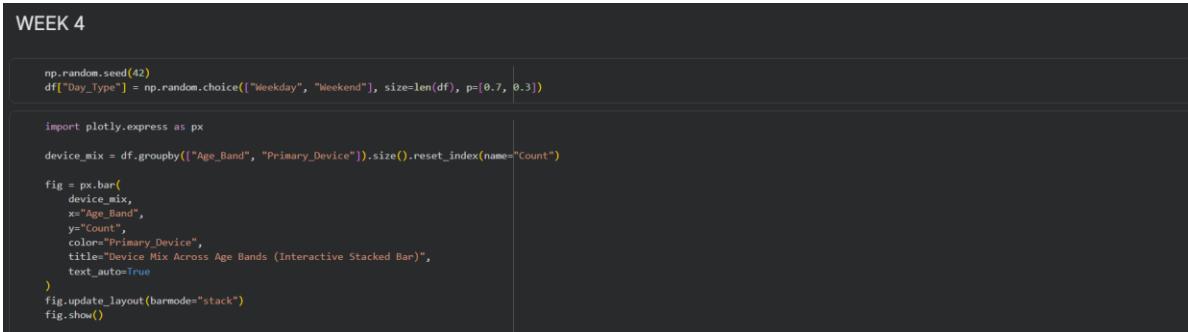
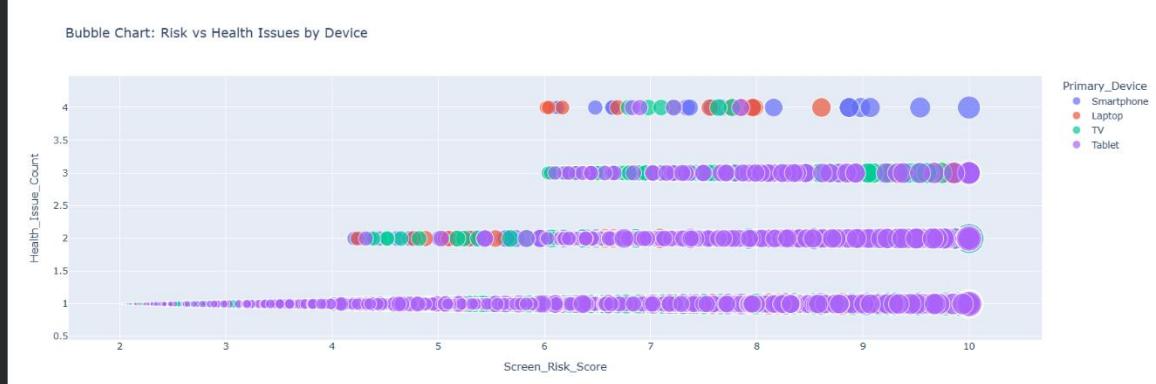
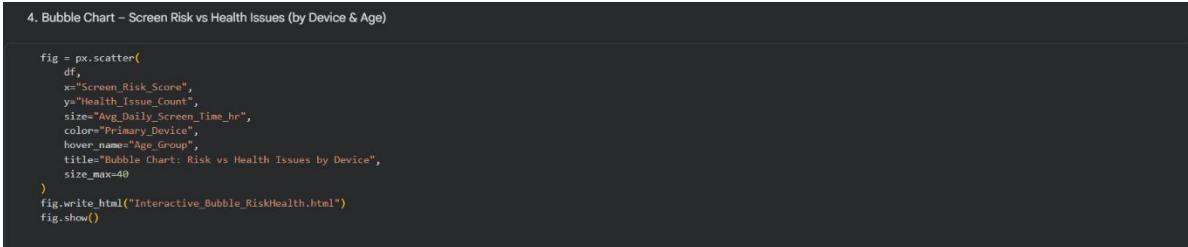
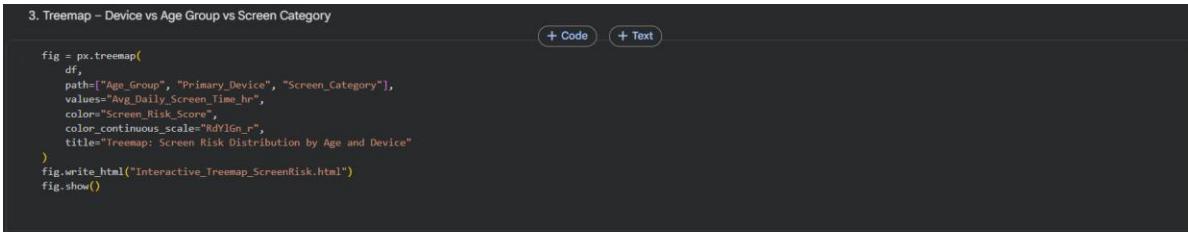
```

plt.figure(figsize=(7,5))
sns.regplot(x='Avg_Daily_Screen_Time_hr', y='Health_Issue_Count', data=df, scatter_kws={'alpha':0.4})
plt.title('Screen Time vs Number of Health Issues')
plt.xlabel('Avg Daily Screen Time (hrs)')
plt.ylabel('Health Issue Count')
plt.tight_layout()
plt.show()

```





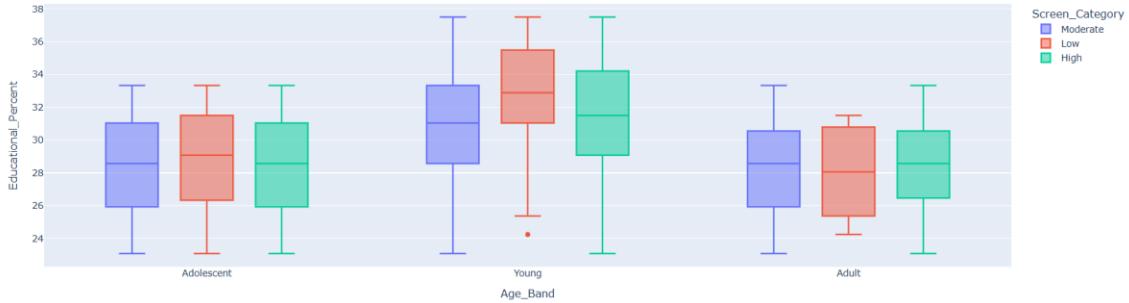


```

fig = px.box(
    df,
    x="Age_Band",
    y="Educational_Percent",
    color="Screen_Catagory",
    title="Educational Usage Percentage by Age Band and Screen Category"
)
fig.show()

```

Educational Usage Percentage by Age Band and Screen Category

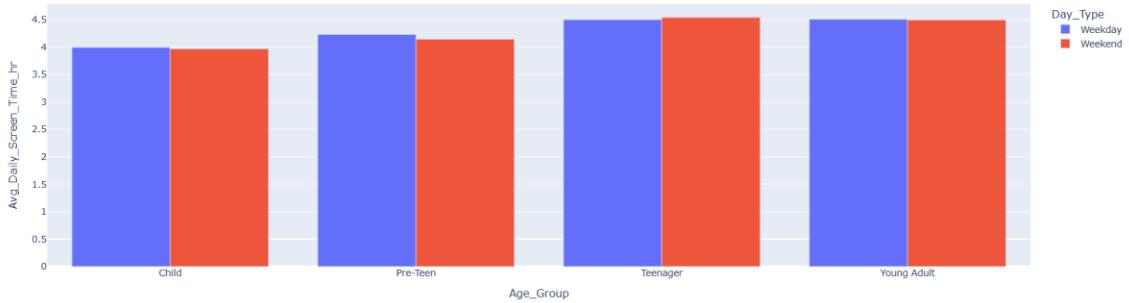


```

fig = px.bar(
    df.groupby(["Day_Type", "Age_Group"])["Avg_Daily_Screen_Time_hr"].mean().reset_index(),
    x="Age_Group",
    y="Avg_Daily_Screen_Time_hr",
    color="Day_Type",
    barmode="group",
    title="Average Screen Time by Age Group: Weekday vs Weekend"
)
fig.show()

```

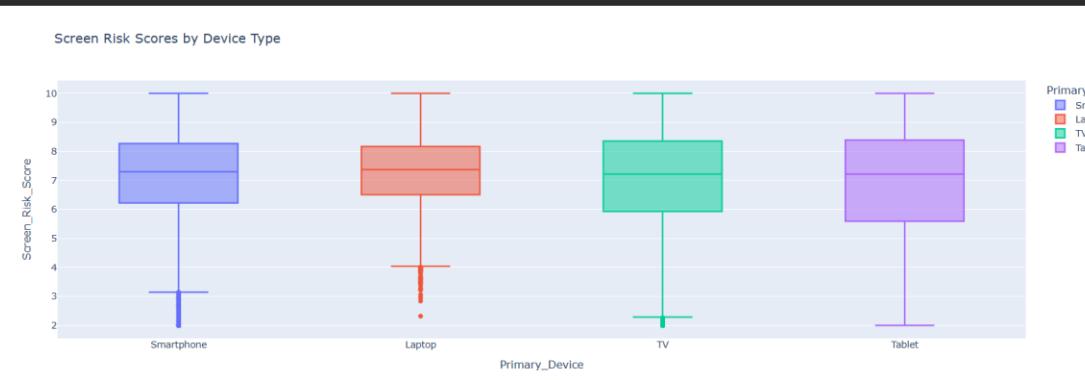
Average Screen Time by Age Group: Weekday vs Weekend



```

fig = px.box(
    df,
    x="Primary_Device",
    y="Screen_Risk_Score",
    color="Primary_Device",
    title="Screen Risk Scores by Device Type",
)
fig.show()

```



```

import seaborn as sns
import matplotlib.pyplot as plt
plt.figure(figsize=(7,4))
sns.heatmap(pd.crosstab(df["Day_Type"], df["Screen_Category"]), annot=True, cmap="YlGnBu", fmt='d')
plt.title("Day Type vs Screen Category (Heatmap)")
plt.show()

```

