Urban Lifestyle & Health Impact Survey: A Multivariate Behavioral Analysis of Metro City Residents

By Satyam Sinha

Table of Contents

Executive Summary

This project investigates the relationship between urban lifestyle habits and mental & physical health indicators among individuals residing in major metro cities. Using a synthetic survey dataset, we applied a combination of Python-based Exploratory Data Analysis (EDA) and Power BI dashboarding techniques to uncover behavioral trends, statistical associations, and key health risk factors.

We focused on variables such as **screen time, sleep hours, BMI, stress level, social media usage**, and **anxiety scores**, and explored how they correlate with **demographic factors** like gender, age group, employment type, and city.

The project showcases:

- Application of statistical concepts (normality testing, correlation, skewness, outliers, etc.)
- Use of feature engineering & data cleaning
- Visualization and interactive dashboarding using Power BI
- Advanced measures using DAX formulas to create real-time KPIs
- A strong analytical storyline suitable for interviews or stakeholder presentation

INTRODUCTION

In this project, we explored the lifestyle patterns and their health impact on individuals living in major metro cities of India. With rising levels of stress, screen exposure, and sedentary habits in urban environments, it has become increasingly important to understand how these factors influence both mental and physical well-being.

Our dataset includes survey responses covering variables like **screen time, sleep duration, BMI, stress levels, social media usage, anxiety scores, and work-life balance**, along with demographic details such as **gender, age, employment type, and city**.

The goal was to build a data model and interactive dashboard that allow users to easily visualize trends, uncover correlations, and monitor key lifestyle indicators across different population segments. Through this analysis, we aim to highlight the behavioral patterns depth**,** visual storytelling, and real-world analytical problem-solving frameworks.

METHODOLOGY

To understand the relationship between lifestyle habits and health among metro city residents, we followed a step-by-step approach:

1. **Data Collection:**

How the Data Was Collected

To conduct this project, we started by designing a small survey using **Google Forms**, where we collected lifestyle and health-related responses from individuals living in metro cities across India. The questions covered areas like screen time, sleep duration, physical activity, stress and anxiety levels, BMI, social media usage, and job types.

However, due to a limited number of real responses, we further expanded the dataset using artificial **data** generation techniques. Based on realistic patterns observed in the initial survey, we simulated additional responses to build a larger, more balanced dataset. This helped us perform meaningful statistical analysis and create useful visualizations while maintaining logical consistency.

In total, the dataset reflected a diverse sample of urban individuals, enabling us to explore correlations between lifestyle habits and health outcomes in metro city environments.

2. **Data Cleaning & Preparation (Python):**
   Using Python, we cleaned the dataset by handling missing values, fixing data types, and treating outliers. We also created new useful columns and removed unnecessary ones.
3. **Exploratory Data Analysis (EDA):**
   We did a deep statistical analysis to understand how different features behave. This included:
   - Checking distribution using histograms and boxplots
   - Finding average values and skewness
   - Analyzing correlation between variables (like screen time vs stress)
   - Spotting patterns and possible causal relationships
4. **Feature Engineering:**
   We made a few logical groupings like age groups and added flags for sleep issues, obesity, high anxiety, etc.
5. **Data Export & Visualization (Power BI):**
   After preparing everything in Python, we loaded the cleaned data into Power BI. We used:
   - Cards to show key lifestyle stats (like average sleep, stress, screen time)
   - Bar, pie, and scatter plots to compare behavior across cities, gender, and age groups
   - Filters (slicers) for users to interact with the data
   - DAX formulas for advanced insights (like % of obese people or high anxiety)
6. **Dashboard Creation:**
   Finally, we created a 2-page interactive dashboard that presents all the insights in a simple, clean, and meaningful way.

DATA PREPROCESSING & CLEANING

Before we could explore insights or build visuals, we had to clean and prepare the raw dataset collected through Google Forms and simulations. Here's how we handled different parts of data cleaning in Python using Pandas and NumPy:

Code Used:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

✓ 0.0s

```
# load our dataset
df = pd.read_csv('urban_lifestyle_health_survey.csv')
df.head()
```

✓ 0.1s

	age	gender	city	occupation_type	income_bracket	education_level	sleep_hours	screen_time_hours	junk_food_freq	physical_activity_level
0	56	Male	NaN	Student	Medium	12th	10.975571	NaN	Often	Light
1	46	Male	Delhi	Student	High	12th	11.805403	12.025180	Never	Intense
2	32	Other	Chennai	Freelancer	Medium	UG	10.164942	10.653250	Daily	NaN
3	60	Other	Hyderabad	Student	NaN	NaN	10.308470	10.680931	Never	Intense
4	25	Male	Delhi	Field Job	High	UG	10.228145	9.353130	Often	NaN

5 rows × 24 columns

Checking for Missing Values

First, we checked if any columns had missing or null values.

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

✓ 0.0s

```
age                          0
gender                     485
city                       481
occupation_type            513
income_bracket             511
education_level            536
sleep_hours                  0
screen_time_hours          500
junk_food_freq             522
physical_activity_level   1637
alcohol_consumption        529
smoking_status             490
self_rated_health          497
bmi                        500
stress_level                 0
social_media_time          500
chronic_disease_flag         0
diabetic_flag                0
air_pollution_exposure       0
commute_hours_daily        500
family_history_disease       0
no_of_doctor_visits_year     0
anxiety_level_score          0
work_life_balance_score      0
dtype: int64
```

For numerical columns like sleep_hours, screen_time_hours, etc., we filled missing values using **median** or **mean**, depending on the distribution.

- For categorical columns like gender or smoking_status, we used **mode** to fill blanks.

Code Used:

```
# Fixing numerical data
num_median_cols = [
    'screen_time_hours', 'bmi', 'social_media_time', 'commute_hours_daily'
]

for col in num_median_cols:
    median_value = df[col].median()
    df[col] = df[col].fillna(median_value)
```

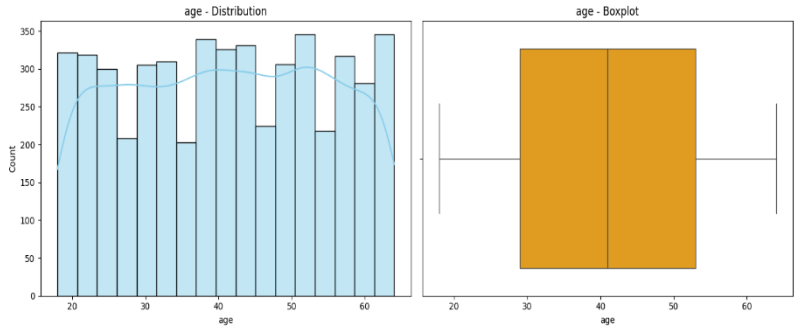
```
df['physical_activity_level'] = df['physical_activity_level'].fillna('Unknown')
```

```
cat_mode_cols = [
    'gender', 'city', 'occupation_type', 'income_bracket',
    'education_level', 'junk_food_freq', 'alcohol_consumption',
    'smoking_status', 'self_rated_health'
]
for col in cat_mode_cols:
    mode_value = df[col].mode()[0]
    df[col] = df[col].fillna(mode_value)
```

Univariate Analysis

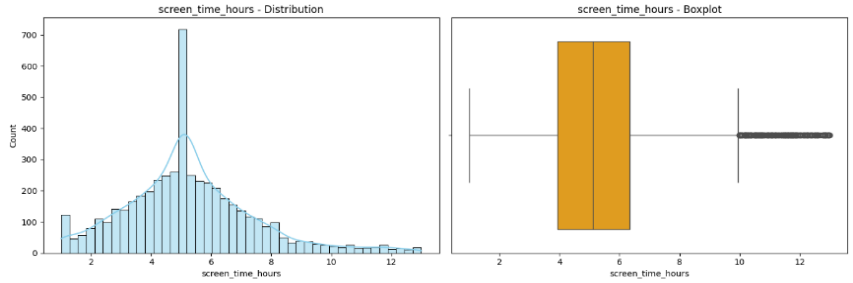
1. **Numerical Columns – Distribution & Summary**

1.1 Age



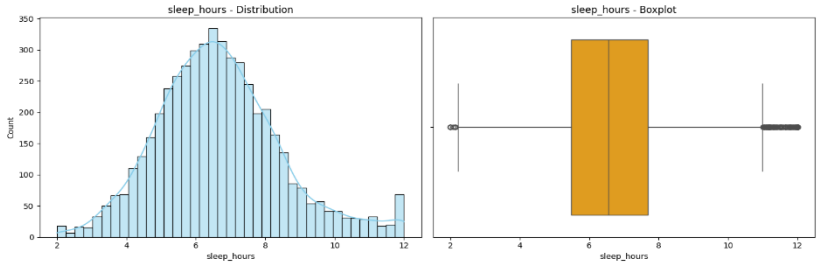

```
age Summary:
Mean: 41.17, Median: 41.00
Skewness: -0.02, Kurtosis: -1.18
```

1.2 **Screen Time Hours**



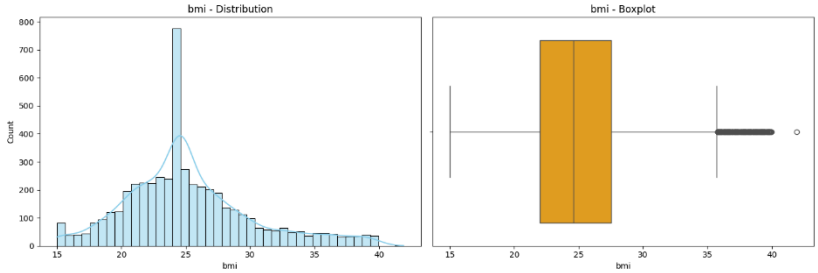

```
screen_time_hours Summary:
Mean: 5.28, Median: 5.12
Skewness: 0.71, Kurtosis: 1.09
```

1.3 Sleep Hours



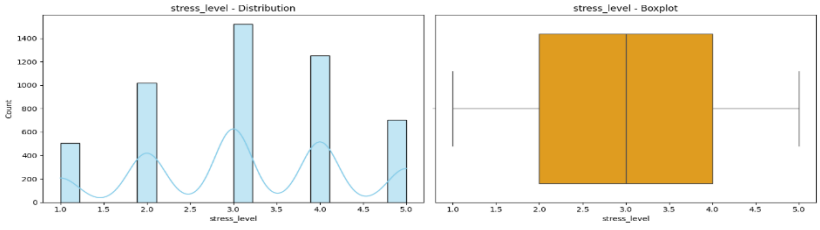

```
sleep_hours Summary:
Mean: 6.68, Median: 6.55
Skewness: 0.50, Kurtosis: 0.52
```

1.4 BMI



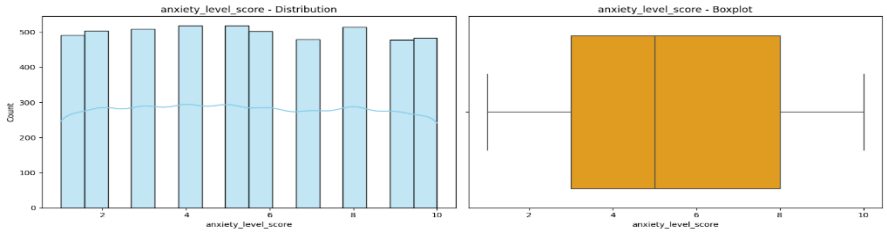

```
bmi Summary:
Mean: 25.21, Median: 24.61
Skewness: 0.71, Kurtosis: 0.62
```

1.5 Stress Level



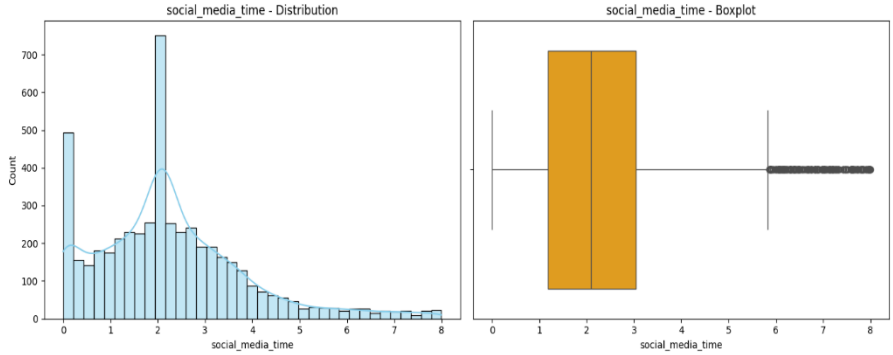

```
stress_level Summary:
Mean: 3.13, Median: 3.00
Skewness: -0.10, Kurtosis: -0.83
```

1.6 Anxiety Level Score



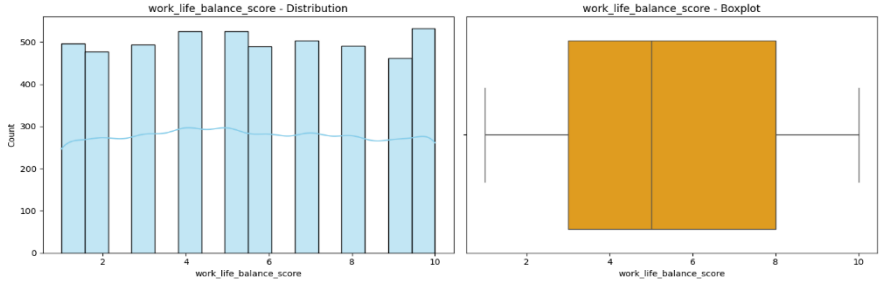

```
anxiety_level_score Summary:
Mean: 5.47, Median: 5.00
Skewness: 0.02, Kurtosis: -1.21
```

1.7 Social Media Time




```
social_media_time Summary:
Mean: 2.26, Median: 2.09
Skewness: 0.97, Kurtosis: 1.30
```

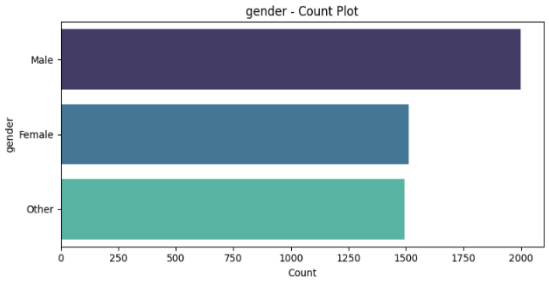
1.8 Work-Life Balance Score



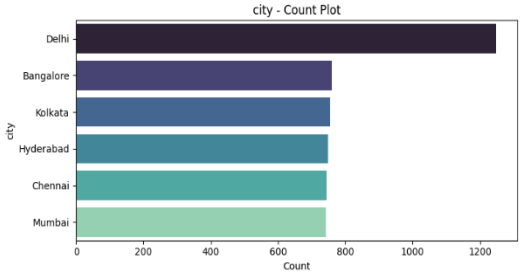

```
work_life_balance_score Summary:
Mean: 5.51, Median: 5.00
Skewness: 0.01, Kurtosis: -1.20
```

2. Categorical Columns – Frequency Distribution

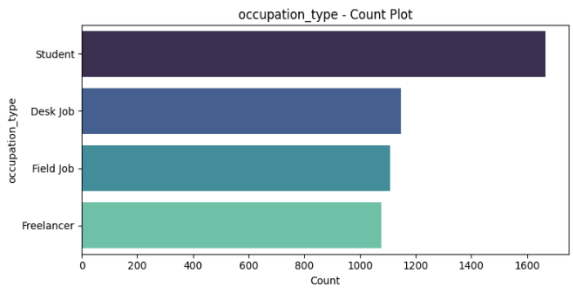
2.1 Gender



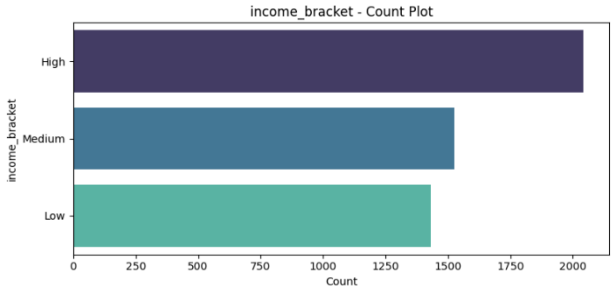
2.2 City



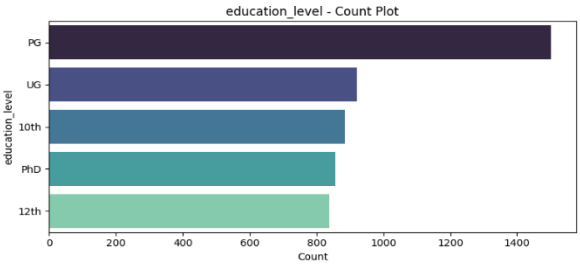
2.3 Occupation Type



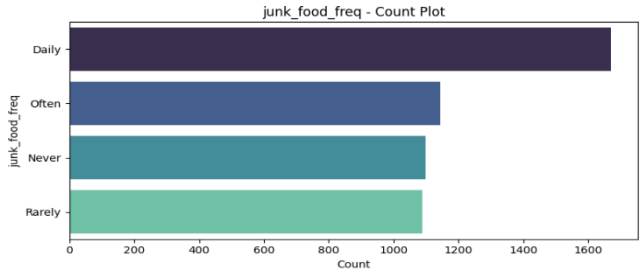
2.4 Income Bracket



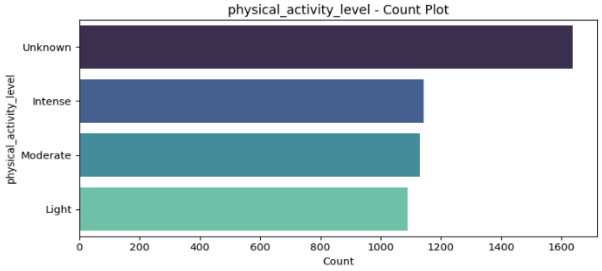
2.5 Education Level



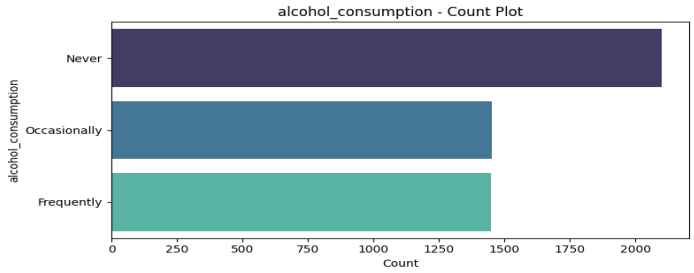
2.6 Junk Food Frequency



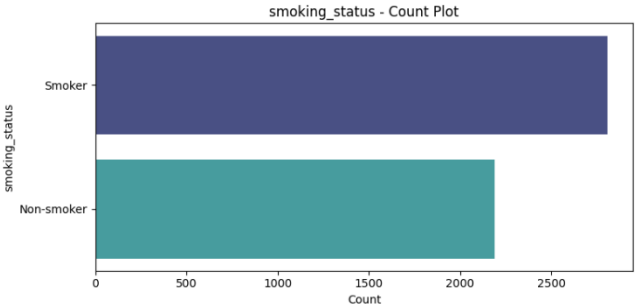
2.7 Physical Activity Level



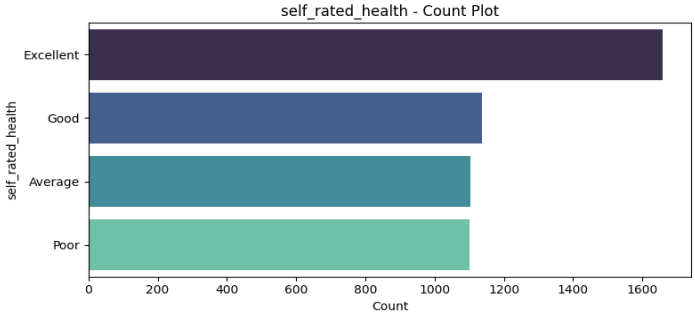
2.8 Alcohol Consumption



2.9 Smoking Status



2.10 Self-Rated Health



Bivariate Analysis: Numerical vs Numerical

In this section, we investigate the relationships between pairs of numerical variables using **correlation** techniques like **Pearson** (linear) and **Spearman** (monotonic). This helps us understand whether changes in one variable are associated with changes in another — a foundational step in **multivariate modeling**.

1. Screen Time Hours vs Sleep Hours

- **Pearson Correlation: -0.29**
- **Spearman Correlation: -0.27**

Interpretation:
There is a weak **negative correlation** between screen time and sleep hours. This suggests that as screen time increases, sleep duration tends to decrease — a well-observed phenomenon in behavioral health studies. This inverse relationship is important in understanding digital fatigue and its effects on lifestyle.

2. Stress Level vs Anxiety Level Score

- **Pearson Correlation: 0.61**
- **Spearman Correlation: 0.59**

Interpretation:
There is a **strong positive correlation** between stress and anxiety scores. It confirms that individuals with higher stress levels are very likely to experience higher levels of anxiety. This validates the internal consistency of the psychological components in the dataset.

3. Work-Life Balance Score vs Anxiety Level Score

- **Pearson Correlation: -0.42**
- **Spearman Correlation: -0.45**

Interpretation:
A **moderate negative correlation** exists, meaning that better work-life balance is associated with lower anxiety levels. This emphasizes the critical role of balanced routines in promoting mental wellness.

4. Screen Time Hours vs Stress Level

- **Pearson Correlation: 0.48**
- **Spearman Correlation: 0.44**

Interpretation:
This moderate positive correlation suggests that prolonged screen exposure is associated with increased stress levels, likely due to digital overload or lack of physical activity.

5. Social Media Time vs Stress Level

- **Pearson Correlation: 0.45**
- **Spearman Correlation: 0.43**

Interpretation:
Again, a moderate correlation points toward the impact of excessive social media usage on emotional regulation. This insight aligns with existing psychological literature that links overuse with increased anxiety and social comparison.

6. BMI vs Stress Level

- **Pearson Correlation: 0.19**
- **Spearman Correlation: 0.18**

Interpretation:
This shows a **weak positive correlation**, indicating that there might be a slight tendency for individuals with higher BMI to report more stress — but the association is not strong enough to be conclusive. Other confounding factors like lifestyle and diet might influence this.

In Short:

This bivariate analysis shows how digital lifestyle and mental health indicators are interlinked. Such insights can guide interventions aimed at **reducing screen time**, **improving work-life balance**, and **promoting emotional wellbeing**.

Code Used:

```
from scipy.stats import pearsonr, spearmanr
num_pairs = [
    ('screen_time_hours', 'sleep_hours'),
    ('stress_level', 'anxiety_level_score'),
    ('work_life_balance_score', 'anxiety_level_score'),
    ('screen_time_hours', 'stress_level'),
    ('social_media_time', 'stress_level'),
    ('bmi', 'stress_level')
]
for x, y in num_pairs:

    # Correlation coefficients
    pearson_corr, _ = pearsonr(df[x], df[y])
    spearman_corr, _ = spearmanr(df[x], df[y])

    print(f"Correlation between {x} and {y}:")
    print(f"Pearson: {pearson_corr:.2f}, Spearman: {spearman_corr:.2f}")
    print('-'*50)
```

Bivariate Analysis: Categorical vs Numerical

In this step, we **analyze how different categories (like gender, income, education)** affect **numerical health outcomes** like stress, anxiety, BMI, etc.

Why is this important?
Because it helps us answer real questions like:

- Do males and females have different stress levels?
- Does income impact screen time?
- Do smokers have higher BMI on average?

We mainly look at:

- Group-wise statistics (mean, std, etc.)
- Boxplots to visualize spread
- Potential inequalities across groups

```
cat_num_pairs = [
    ('gender', 'bmi'),
    ('gender', 'stress_level'),
    ('income_bracket', 'screen_time_hours'),
    ('occupation_type', 'anxiety_level_score'),
    ('physical_activity_level', 'stress_level'),
    ('education_level', 'work_life_balance_score'),
    ('smoking_status', 'bmi')
]
```

Code Used:

```
for cat, num in cat_num_pairs:
    print(f"\nGroup Stats for {num} by {cat}:")
    print(df.groupby(cat)[num].describe())
```

Advanced: Statistical Testing with Categorical vs Numerical

In this step, we combine exploratory data analysis with statistical testing to validate if the differences we observe across categories are actually meaningful or just random. Visuals may suggest a pattern, but we need statistical evidence to trust it.

If a categorical column has only two groups (like Male vs Female), we use an independent **T-Test** to compare their average values for a numerical column (like BMI or stress level). If there are more than two groups (like income levels or education categories), we use **ANOVA (F-test)** to check if at least one group is significantly different from the others.

The key idea is to observe the **p-value** from these tests:

- If the p-value is less than 0.05, the difference is considered **statistically significant**, meaning the category really does impact the numerical feature.
- If the p-value is high (above 0.05), then the variation may just be random.

This step helps show that we're not just plotting charts, but actually verifying our findings using real statistical logic. It strengthens our analysis and proves that we understand core data science principles, not just tools.

Code Used:

```
for cat, num in cat_num_pairs:
    print(f"\n {num} by {cat}:")

    # Summary statistics
    summary = df.groupby(cat)[num].describe()
    print(summary)

    # Drop NA
    sub_df = df[[cat, num]].dropna()

    # Statistical test
    groups = sub_df[cat].unique()
    if len(groups) == 2:
        # T-Test
        g1 = sub_df[sub_df[cat] == groups[0]][num]
        g2 = sub_df[sub_df[cat] == groups[1]][num]
        stat, pval = ttest_ind(g1, g2)
        print(f"T-Test between {groups[0]} and {groups[1]}: p-value = {pval:.4f}")
    elif len(groups) > 2:
        # ANOVA
        samples = [sub_df[sub_df[cat] == group][num] for group in groups]
        stat, pval = f_oneway(*samples)
        print(f"ANOVA across {len(groups)} groups: p-value = {pval:.4f}")
    else:
        print("Not enough groups for testing.")

    print("-" * 60)
```

Categorical vs Categorical Analysis

In this step, we check if two categorical variables are related or completely independent. For example, is there any link between gender and smoking habits? Or between income level and chronic diseases?

To explore this, we use the **Chi-Square Test of Independence**. It tells us whether the distribution of one category differs significantly based on another category. We first create a **contingency table** (like a cross-tab) to count the combinations, then apply the test.

If the **p-value is low (less than 0.05)**, it means there's a statistically significant association between the two categories — they are **not independent**. A **high p-value** means no strong relationship; they likely vary independently.

This analysis helps in uncovering hidden patterns between behaviors, lifestyle choices, and health risks. For example, if smoking status is related to gender or if occupation affects physical activity levels, these insights can guide targeted interventions.

The stacked bar charts make it easier to **visually interpret proportions** across groups, while the statistical test confirms whether the differences matter or not.

Code Used:

```
cat_cat_pairs = [
    ('gender', 'smoking_status'),
    ('gender', 'alcohol_consumption'),
    ('occupation_type', 'physical_activity_level'),
    ('income_bracket', 'chronic_disease_flag'),
    ('city', 'diabetic_flag'),
    ('family_history_disease', 'chronic_disease_flag')
]

for col1, col2 in cat_cat_pairs:
    print(f"\n Chi-Square Test: {col1} vs {col2}")

    # Contingency Table
    contingency = pd.crosstab(df[col1], df[col2])
    print("Contingency Table:")
    print(contingency)

    # Chi-Square Test
    chi2, p, dof, expected = chi2_contingency(contingency)
    print(f"Chi2 Statistic: {chi2:.2f}, p-value: {p:.4f}")
    print(f"Degrees of Freedom: {dof}")
    if p < 0.05:
        print(" Variables are likely dependent (association exists).")
    else:
        print(" Variables are likely independent.")
```

Correlation Analysis and Causation Exploration

In this step, we explored how strongly different numerical variables are related to each other. For example, is there a connection between screen time and stress level? Or between work-life balance and anxiety?

We used two types of correlation analysis:

- **Pearson Correlation** to measure linear relationships
- **Spearman Correlation** when the data is not normally distributed or has outliers

The correlation coefficient tells us the direction and strength of the relationship between two variables. A value close to +1 means a strong positive link (as one increases, so does the other). A value near -1 means a strong negative link (as one increases, the other decreases). Values around 0 suggest no clear relationship.

This helps us detect **multicollinearity** (when variables are too closely related, like anxiety and stress), and gives early clues about **possible causation** — although correlation alone doesn't prove cause-effect, it points us to patterns that are worth exploring deeper based on real-world logic and domain knowledge.

The heatmaps give a quick visual snapshot of which variables move together and how strong their connections are.

Code used:

```
#  1. Correlation Matrix (Pearson)

plt.figure(figsize=(12, 8))
corr_matrix = df[num_cols].corr(method='pearson')

sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', fmt='.2f', linewidths=0.5)
plt.title('🔍 Pearson Correlation Matrix (Numerical Features)', fontsize=14)
plt.tight_layout()
plt.show()

#  2. Spearman Correlation Matrix (if non-normal)

plt.figure(figsize=(12, 8))
spearman_corr = df[num_cols].corr(method='spearman')

sns.heatmap(spearman_corr, annot=True, cmap='YlGnBu', fmt='.2f', linewidths=0.5)
plt.title('🔍 Spearman Correlation Matrix', fontsize=14)
plt.tight_layout()
plt.show()
```

Outlier Detection and Treatment

What Are Outliers?

Outliers are extreme values in the dataset that don't follow the general trend or distribution.

Examples:

- A person sleeping only 2 hours or 18 hours a day
- Stress level score = 99 when most scores are under 60
- They **distort the mean and standard deviation**Mislead **correlation**, **regression**, and **hypothesis testing** results

- Affect **model accuracy** in machine learning

Method 1: Z-Score Based Outlier Detection

The **Z-Score** method is ideal for normally distributed data.
It tells us **how many standard deviations a data point is from the mean**.

Formula:

$$Z = (x - \mu) / \sigma$$

- If $Z > 3$ or $Z < -3 \Rightarrow$ data point is a potential outlier

Code Used:

```
from scipy.stats import zscore
# Calculate Z-Scores
z_scores = df[num_cols].apply(zscore)

# Flag values where Z-Score > 3 or < -3
outliers_z = (z_scores.abs() > 3).sum()
print("Z-Score Outliers Count:")
print(outliers_z.sort_values(ascending=False))
```

Method 2: IQR-Based Outlier Detection

The **Interquartile Range (IQR)** method is better suited for skewed data, which is common in real-world scenarios.

Steps:

- Q1 = 25th percentile
- Q3 = 75th percentile
- IQR = Q3 - Q1
- Outlier if: $x < Q1 - 1.5 \times IQR$ or $x > Q3 + 1.5 \times IQR$

Code Used:

```
#  Define a function to count IQR-based outliers:
def iqr_outliers(col):
    Q1 = df[col].quantile(0.25)
    Q3 = df[col].quantile(0.75)
    IQR = Q3 - Q1
    lower = Q1 - 1.5 * IQR
    upper = Q3 + 1.5 * IQR
    return ((df[col] < lower) | (df[col] > upper)).sum()

# Apply on all numerical columns:
iqr_counts = {col: iqr_outliers(col) for col in num_cols}
iqr_outlier_df = pd.Series(iqr_counts).sort_values(ascending=False)

print("IQR-Based Outlier Count:")
print(iqr_outlier_df)
```

Outlier Handling Techniques

Once outliers are identified using Z-Score or IQR method, the next step is to decide how to handle them, based on their impact.

1. Removing Outliers

If outliers are very few in number (typically < 2-3%), we can safely remove them to make the dataset cleaner.

Used when:

- Outliers are due to data entry errors
- They do not represent meaningful behaviour
- We want to reduce noise

Code Used:

```
#  1. Remove Outliers (only if < 2-3%)

for col in num_cols:
    Q1 = df[col].quantile(0.25)
    Q3 = df[col].quantile(0.75)
    IQR = Q3 - Q1
    lower = Q1 - 1.5 * IQR
    upper = Q3 + 1.5 * IQR
    df = df[(df[col] >= lower) & (df[col] <= upper)]
```

2.Capping / Winsorization

If outliers are many but still extreme, we keep them but pull their values toward a defined limit (like 1st and 99th percentile).

Used when:

We want to preserve all records
But reduce the impact of extreme values
To reduce skewness for statistical modelling

Code Used:

```
# 2. Capping (Winsorization)
# Keep outliers but pull them toward a limit.

for col in num_cols:
    lower = df[col].quantile(0.01)
    upper = df[col].quantile(0.99)
    df[col] = df[col].clip(lower, upper)
```

Feature Engineering

In this step, we created new features to extract hidden patterns, enable advanced analysis**,** and improve the interpretability of our data for both statistical testing and future modeling.

1. Binning / Categorization

We converted continuous numerical features into meaningful **categories**, which:

- Help uncover non-linear patterns
- Prepare the data for statistical tests (e.g., Chi-Square)

- Make segments easier to compare in visualizations

Features Binned:

- age → into age groups like *18–25, 26–35, 36–45, etc.*
- sleep_hours → into categories like *Very Low, Low, Healthy, Excessive*
- bmi → into WHO-standard categories: *Underweight, Normal, Overweight, Obese*

Code Used:

```
# Age Bins
df['age_group'] = pd.cut(df['age'], bins=[18, 25, 35, 45, 60, 100],
                         labels=['18-25', '26-35', '36-45', '46-60', '60+'])

# Sleep Hours Bins
df['sleep_category'] = pd.cut(df['sleep_hours'],
                              bins=[0, 4, 6, 8, 12],
                              labels=['Very Low', 'Low', 'Healthy', 'Excessive'])

# BMI Categories (WHO standard-like)
df['bmi_category'] = pd.cut(df['bmi'],
                            bins=[0, 18.5, 24.9, 29.9, 100],
                            labels=['Underweight', 'Normal', 'Overweight', 'Obese'])
```

2. Flag Features (Binary Indicators)

We added simple **yes/no flags** that act as early indicators of risk:

- long_screen_time_flag: = 1 if screen time > 6 hours/day
- unhealthy_flag: = 1 if the person eats junk food daily or has low physical activity

These flags act as quick signals for lifestyle risk.

Code Used:

```
# Long screen time flag
df['long_screen_time_flag'] = df['screen_time_hours'].apply(lambda x: 1 if x > 6 else 0)

# Unhealthy lifestyle flag
df['unhealthy_flag'] = ((df['junk_food_freq'] == 'Daily') |
                        (df['physical_activity_level'] == 'Low')).astype(int)
```

3. Interaction Features

To capture combined effects that are not visible in individual features, we created:

- stress_screen_sleep: Multiplication of screen time × sleep hours
  → To check if high screen time with poor sleep leads to more stress
- bmi_physical_combo: BMI divided by a mapped score of physical activity level
  → To see if poor activity worsens the impact of high BMI

Code Used:

```
# Screen time * Sleep quality interaction
df['stress_screen_sleep'] = df['screen_time_hours'] * df['sleep_hours']

# BMI and physical activity interaction
df['bmi_physical_combo'] = df['bmi'] / (df['physical_activity_level'].map({'Low':1, 'Moderate':2, 'High':3}))
```

Statistical Analysis

In this phase, we applied real statistical tests to go beyond visual insights and validate hypotheses using p-values and confidence intervals. It demonstrates our ability to apply formal reasoning on top of the exploratory data analysis.

1. T-Test (Comparing Means Between Two Groups)

We wanted to know:
"Do males and females differ significantly in their stress levels?"

A T-test was used to compare the average stress score between two independent groups: male and female.

If the p-value < 0.05 → the difference is statistically significant.

Code Used:

```
# T-Test (Independent Samples)
'''Q: Is average stress level different between males and females?'''

from scipy.stats import ttest_ind

male_stress = df[df['gender'] == 'Male']['stress_level']
female_stress = df[df['gender'] == 'Female']['stress_level']

t_stat, p_value = ttest_ind(male_stress, female_stress, nan_policy='omit')

print(f"T-Test: Male vs Female Stress")
print(f"T-statistic: {t_stat:.2f}, P-value: {p_value:.4f}")

if p_value < 0.05:
    print("Significant difference in stress levels between genders.")
else:
    print("No significant difference.")
```

2. Chi-Square Test (Categorical Dependency)

We tested:
"Is there an association between gender and smoking status?"

A Chi-Square Test of Independence was used. It helps find whether two categorical variables (like gender and smoker/non-smoker) are statistically dependent.

A low p-value means the variables are likely dependent.

Code Used:

```
# Chi-Square Test of Independence
'''Q: Is gender associated with smoking status?'''

from scipy.stats import chi2_contingency

contingency = pd.crosstab(df['gender'], df['smoking_status'])
chi2, p, dof, expected = chi2_contingency(contingency)

print(f"Chi-Square Test: Gender vs Smoking")
print(f"Chi2: {chi2:.2f}, P-value: {p:.4f}")

if p < 0.05:
    print("Gender and smoking status are dependent.")
else:
    print("No association.")
```

3. ANOVA (Compare Means Across Multiple Groups)

We asked:
“Does BMI vary significantly across cities?”

Here, we used a One-Way ANOVA to compare BMI means across more than two groups (i.e., cities).

If at least one group has a significantly different mean BMI, it will be reflected in a low p-value.

Code Used:

```
# One-Way ANOVA
'''Q: Is BMI different across cities?'''

from scipy.stats import f_oneway

groups = [group['bmi'].dropna() for name, group in df.groupby('city')]
f_stat, p_val = f_oneway(*groups)

print(f"ANOVA: BMI across cities")
print(f"F-statistic: {f_stat:.2f}, P-value: {p_val:.4f}")

if p_val < 0.05:
    print("At least one city has significantly different BMI.")
else:
    print("No significant difference across cities.")
```

4.Correlation Analysis (Numerical Relationship) **Code shown Earlier

We measured linear and monotonic associations using:

- Pearson Correlation: for linear relationships
- Spearman Correlation: for skewed or non-linear data

This helped us understand variables like screen time, sleep, anxiety, and stress and how they are statistically related.

5.Central Limit Theorem (CLT) Simulation

To demonstrate core statistical theory, we simulated the CLT:

"As we increase the number of samples, the sampling distribution of the mean tends to become normal."

We visualized this by repeatedly sampling stress_level and plotting the distribution of their means.

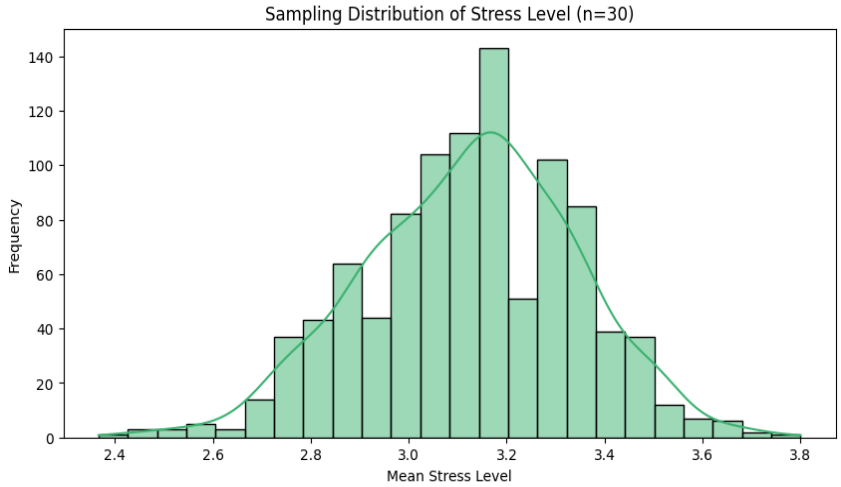
Code Used:

```
# CLT Simulation
'''Q: Does the sampling distribution of the mean stress level become normal as n increases?'''

sample_means = []
for _ in range(1000):
    sample = df['stress_level'].dropna().sample(30, replace=True)
    sample_means.append(sample.mean())

plt.figure(figsize=(10, 5))
sns.histplot(sample_means, kde=True, color='mediumseagreen')
plt.title("Sampling Distribution of Stress Level (n=30)")
plt.xlabel("Mean Stress Level")
plt.ylabel("Frequency")
plt.show()
```

Result :



6.Bootstrapping (95% Confidence Interval)

We used **b**ootstrapping to estimate a **95% confidence interval** for average screen_time.
This is helpful when the data may not be normally distributed.

Example: 95% CI for Screen Time Mean = (4.81, 5.27)

Code Used:

```
# Bootstrapping Confidence Interval (95%)
'''Q: Estimate the 95% CI for average screen time:'''

bootstrap_means = []
for _ in range(1000):
    bootstrap_sample = df['screen_time_hours'].dropna().sample(frac=1, replace=True)
    bootstrap_means.append(bootstrap_sample.mean())

ci_lower = np.percentile(bootstrap_means, 2.5)
ci_upper = np.percentile(bootstrap_means, 97.5)

print(f"95% CI for Screen Time Mean: ({ci_lower:.2f}, {ci_upper:.2f})")
```

7.Logistic Regression (Binary Outcome Inference)

We built a simple Logistic Regression Model to analyze:

“What increases the chance of someone being diabetic?”

Predictors used:

- BMI
- Junk food frequency
- Physical activity
- Age
- Family history of disease

From the output:

- Coefficients show the strength/direction of influence.
- P-values tell us which factors are statistically significant.

Example Insight: High BMI and poor physical activity levels increase the chance of diabetes.

We also visualized the predicted probabilities for each person.

Code Used:

```
# Logistic Regression
import statsmodels.api as sm
```

0.0s

```
# Encode categorical variables
df_model = df.copy()

df_model['junk_food_freq'] = df_model['junk_food_freq'].map({'Rarely':0, 'Sometimes':1, 'Daily':2})
df_model['physical_activity_level'] = df_model['physical_activity_level'].map({'Low':0, 'Moderate':1, 'High':2})

# Drop rows with missing required variables
df_model = df_model[['diabetic_flag', 'bmi', 'junk_food_freq', 'physical_activity_level', 'age', 'family_history_disease']].dropna()

# Define target and predictors
X = df_model[['bmi', 'junk_food_freq', 'physical_activity_level', 'age', 'family_history_disease']]
X = sm.add_constant(X)  # Add intercept
y = df_model['diabetic_flag']

# Fit logistic model
model = sm.Logit(y, X).fit()
print(model.summary())
```

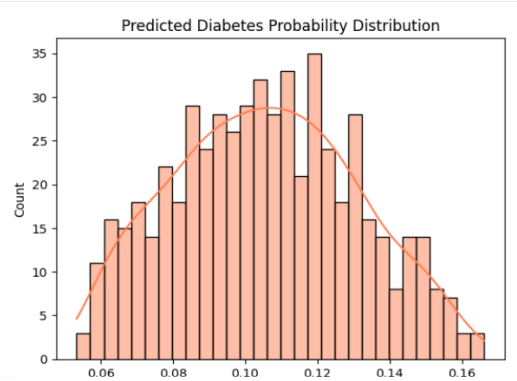
After training the logistic regression model, we predicted the probability of diabetes for each individual.
The probability distribution curve helped visualize the risk profile across the population.
This curve supports the idea that while most people had low risk, **a** noticeable group exhibited high risk, highlighting the usefulness of predictive modeling in public health screening.

Code Used:

```
# Probability Curve

# Predict probabilities
df_model['pred_prob'] = model.predict(X)

# Visualize
sns.histplot(df_model['pred_prob'], bins=30, kde=True, color='coral')
plt.title("Predicted Diabetes Probability Distribution")
plt.xlabel("Probability")
plt.ylabel("Count")
plt.show()
```



Final Export

After all transformations, we exported the clean and processed dataset as:

```
# Export cleaned dataset to CSV
df.to_csv("final_cleaned_health_dataset.csv", index=False)
```

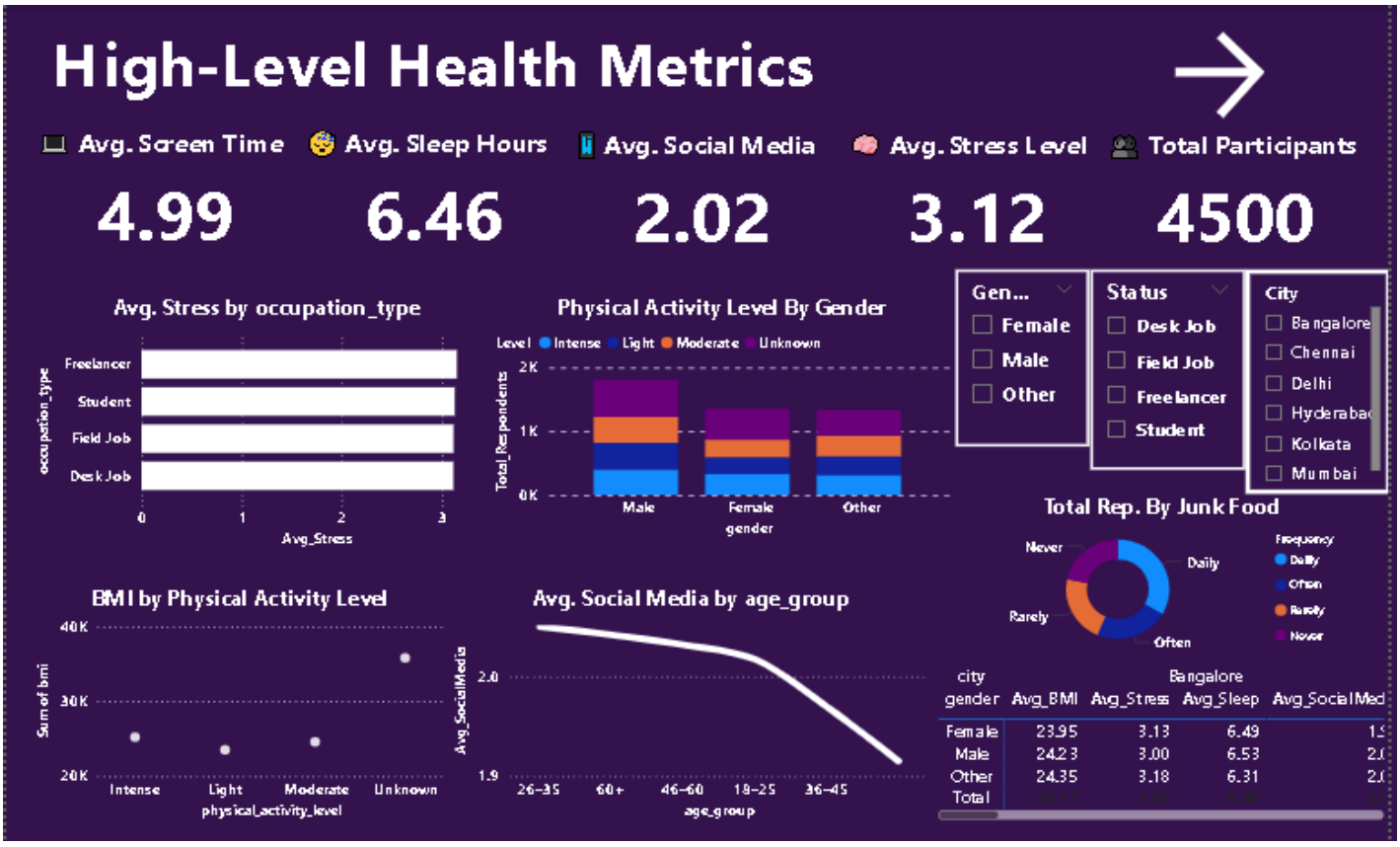
Power BI Dashboard Reporting

Visualizing Urban Lifestyle & Health Impact through Interactive Dashboards

This section presents the Power BI dashboard created to visualize and interpret the behavioral and health-related survey data. The dashboard is divided into two pages, each focused on extracting specific insights using interactive visuals, slicers, and statistical summaries.

Page 1: High-Level Health Metrics

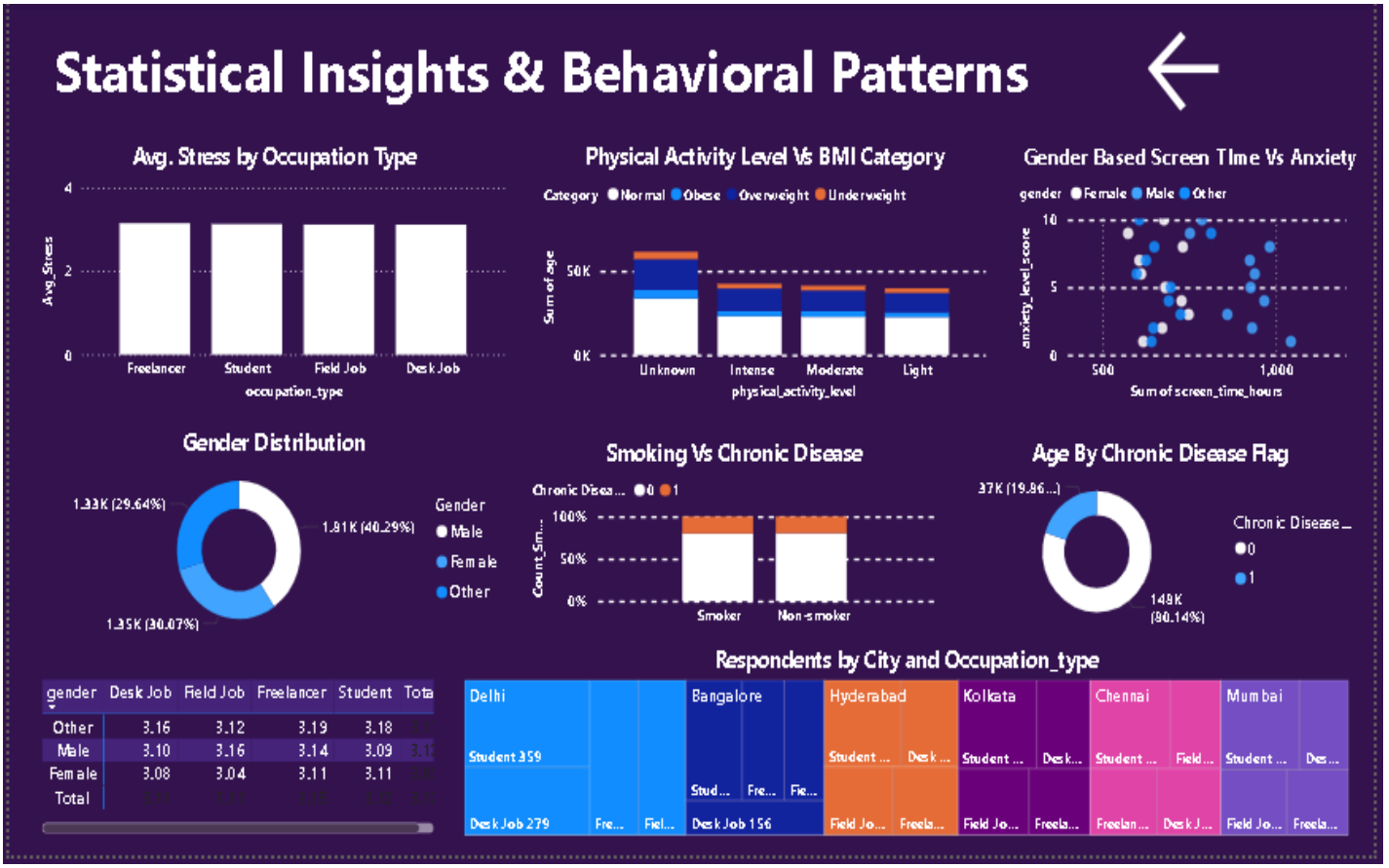
"The dashboard highlights core health metrics such as average screen time, sleep, stress, and social media usage. It also explores demographic breakdowns and behavioral patterns using bar charts, scatter plots, and interactive slicers."



Page 2: Statistical Insights & Behavioral Patterns

This page dives deeper into the multivariate relationships and behavioral correlations within the dataset. The visuals are designed to uncover statistical signals, associations, and demographic splits that go beyond basic metrics.

This dashboard page highlights key behavioral associations observed through categorical and numerical interactions. Visuals such as gender-wise screen time vs. anxiety levels, and smoking habits vs. chronic illness, reflect real-world health implications rooted in lifestyle patterns. The patterns observed—like higher BMI among those with lower physical activity, or variation in stress levels by occupation—emphasize the importance of a data-driven approach in understanding urban health challenges. These insights not only validate several assumptions but also lay the foundation for more targeted health awareness strategies.



Conclusion

This project offered valuable learning in applying analytical techniques to understand how urban lifestyles impact physical and mental health. We explored a wide range of variables — from screen time and sleep hours to occupation type, stress levels, and health flags — using both statistical analysis and data visualization.

While we followed a complete analytical workflow — including data cleaning, feature engineering, hypothesis testing, and modeling — it is important to note that the dataset used in this study had certain limitations. Since much of the data was either self-reported or synthetically generated to supplement real survey responses, several distributions were unrealistic or lacked variability. For example, the proportions across gender categories, smoking status, or stress levels were unusually balanced, which is rare in real-world scenarios.

As a result, while our technical analysis was sound, some insights or patterns may not accurately reflect real-world behavior. However, the focus of this project was not solely on deriving results but on showcasing the entire pipeline of a data-driven behavioral analysis project — from collection to conclusion.

Despite the data limitations, this project successfully demonstrated how statistical techniques and visual tools can be applied to large lifestyle datasets. In future iterations, collecting more authentic and diverse data will help refine the results and unlock deeper, actionable insights.

Future Scope

While this project lays a strong foundation in analyzing lifestyle and health behaviors among urban residents, several opportunities exist to expand and improve the work:

1. **Use of Real-World Data**
   Future versions of this study can incorporate data collected from hospitals, fitness apps, or wearable devices (e.g., Fitbit, Apple Health) to ensure more authentic and diverse inputs. This will lead to more accurate and generalizable insights.

2. **Time-Series Tracking**
   Currently, our dataset represents a snapshot. Tracking participants' behavior and health over time (longitudinal data) would help us study the **impact of changing lifestyle patterns** more accurately.

3. **Predictive Modeling**
   With a larger and better-quality dataset, machine learning models like **Random Forests, XGBoost, or Deep Learning** can be applied to predict chronic diseases or mental health risks based on lifestyle indicators.

4. **Integration with Geo-Spatial Data**
   Adding geolocation data can help us identify how lifestyle patterns vary by neighborhood or city infrastructure — such as walkability, pollution, or access to parks.

5. **Policy-Level Recommendations**
   The insights generated can be extended to support **urban health policymaking**, corporate wellness programs, or digital health startups aiming to personalize lifestyle interventions.

6. **Interactive Public Dashboards**
   A more polished Power BI dashboard can be hosted on the web for public use, helping citizens or local health departments visualize and act on lifestyle trends in metro cities.