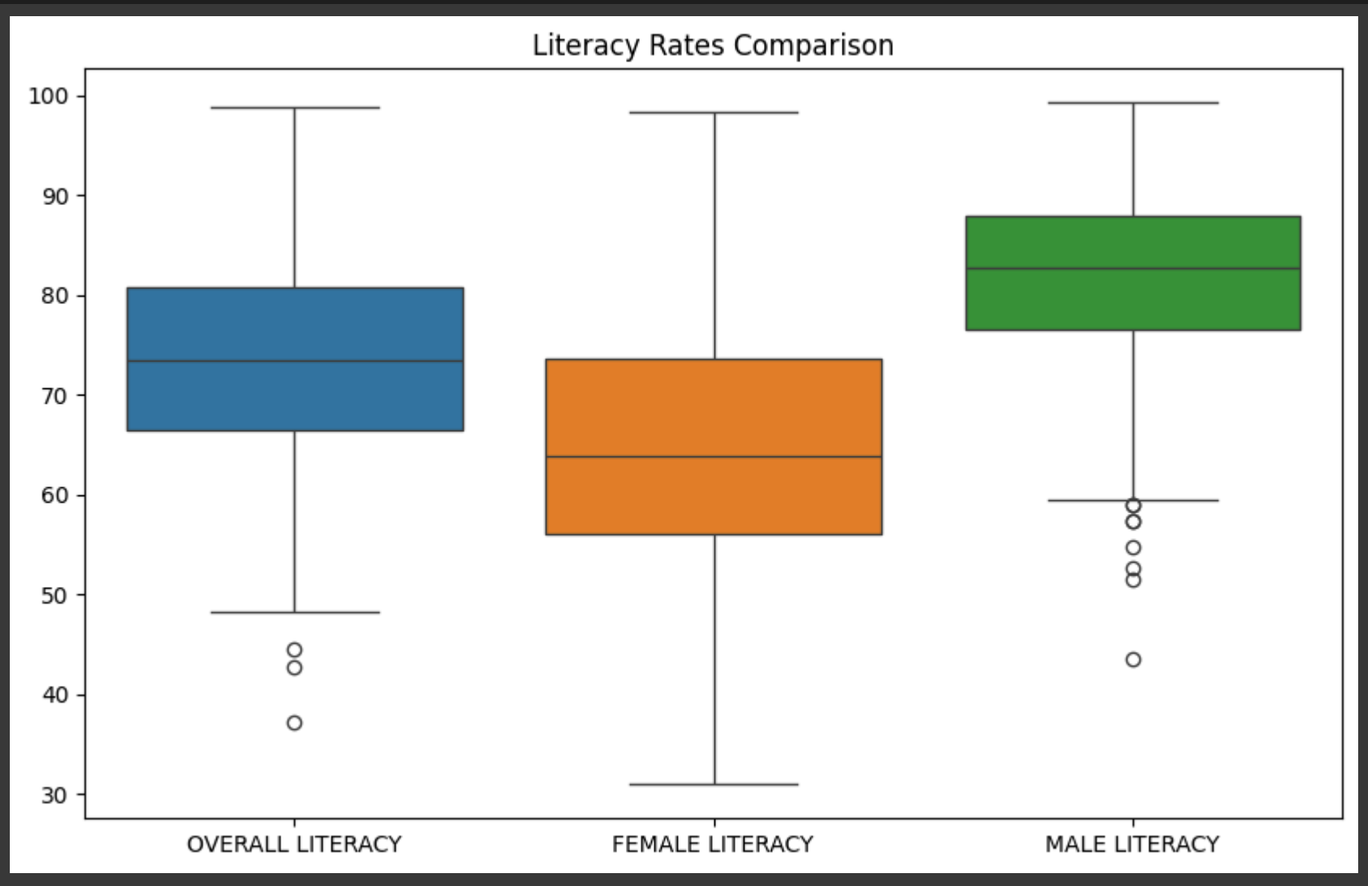
```
!pip install pandas matplotlib seaborn plotly
```

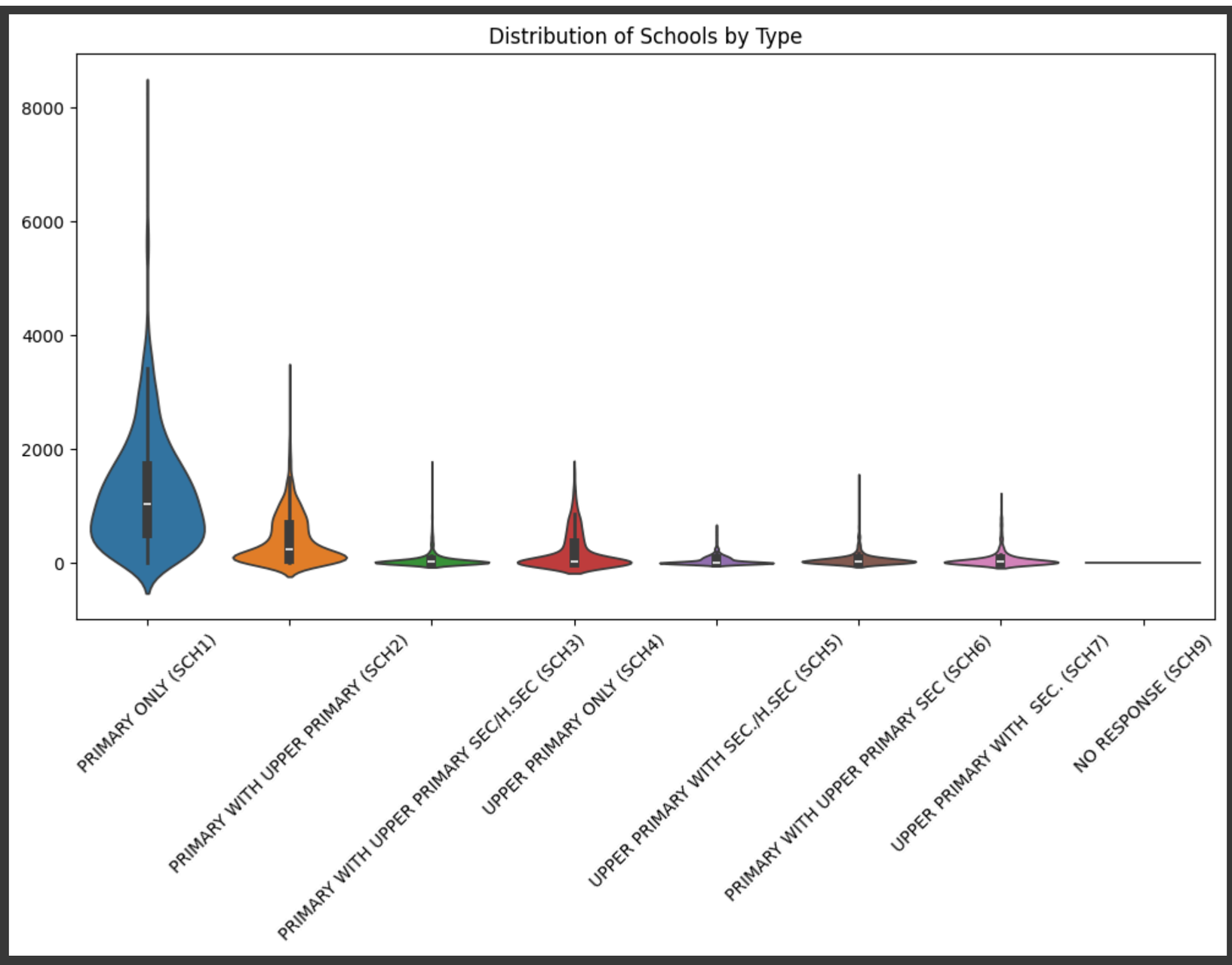
```
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-packages (2.1.4)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist-packages (3.7.1)
Requirement already satisfied: seaborn in /usr/local/lib/python3.10/dist-packages (0.13.1)
Requirement already satisfied: plotly in /usr/local/lib/python3.10/dist-packages (5.15.0)
Requirement already satisfied: numpy<2,>=1.22.4 in /usr/local/lib/python3.10/dist-packages (from pandas) (1.26.4)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.10/dist-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas) (2024.1)
Requirement already satisfied: tzdata>=2022.1 in /usr/local/lib/python3.10/dist-packages (from pandas) (2024.1)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (1.2.1)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (4.53.1)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (1.4.5)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (24.1)
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (9.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (3.1.2)
Requirement already satisfied: tenacity>=6.2.0 in /usr/local/lib/python3.10/dist-packages (from plotly) (9.0.0)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.8.2->pandas) (1.16.0)
```

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

# Load the data
df = pd.read_csv('elementary_2015_16.csv')
```

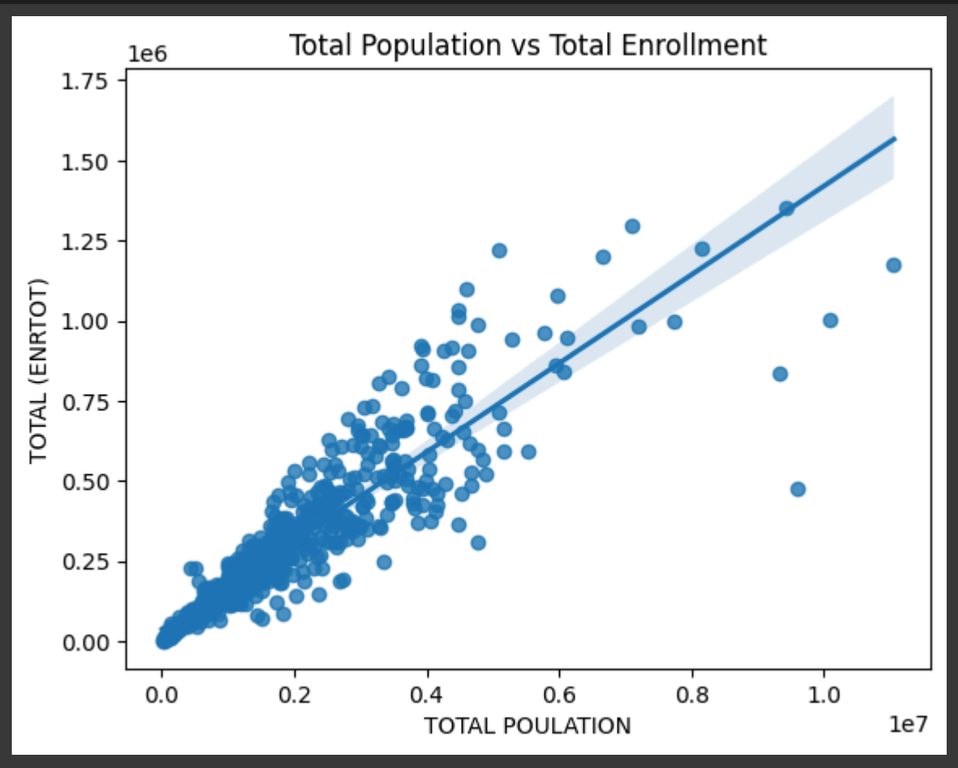
```
plt.figure(figsize=(10, 6))
sns.boxplot(data=df[['OVERALL LITERACY', 'FEMALE LITERACY', 'MALE LITERACY']])
plt.title('Literacy Rates Comparison')
plt.show()
plt.close()
```



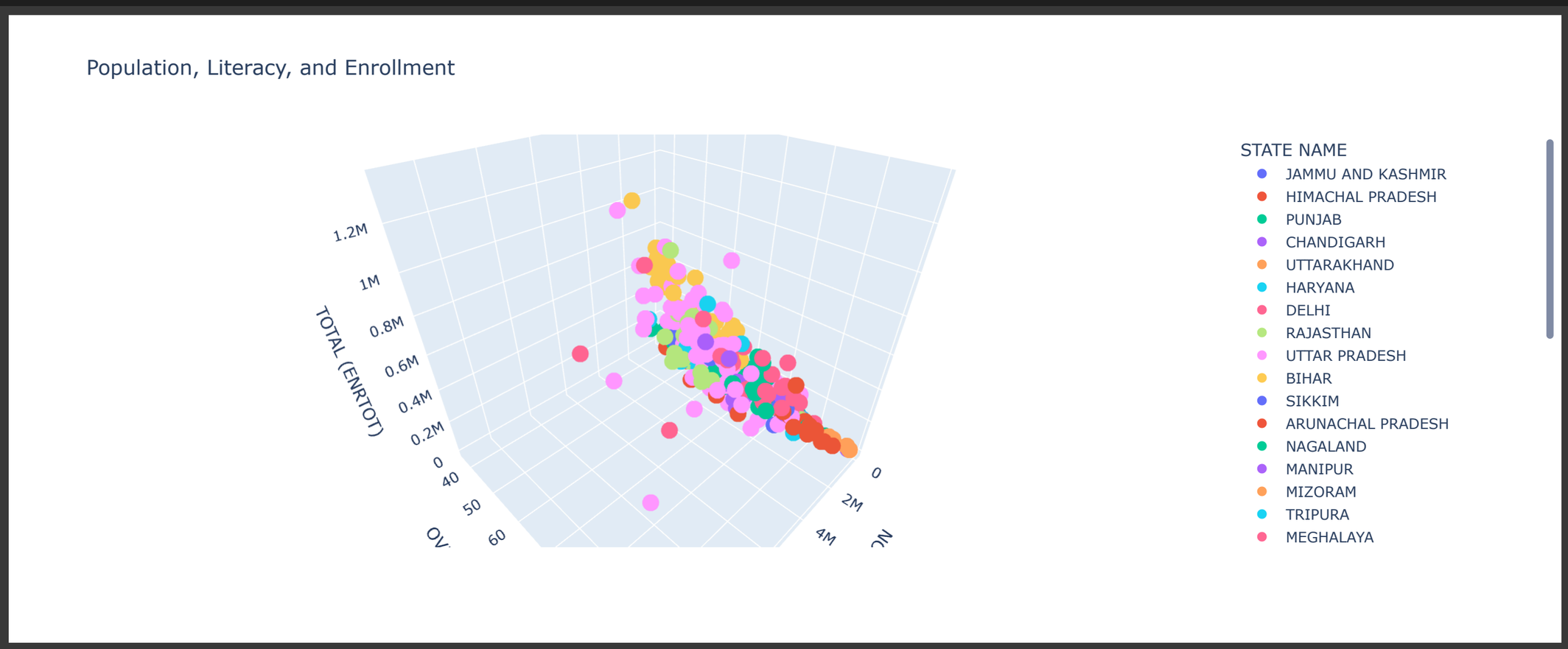

```
school_types = [
  'PRIMARY ONLY (SCH1)',
  'PRIMARY WITH UPPER PRIMARY (SCH2)',
  'PRIMARY WITH UPPER PRIMARY SEC/H.SEC (SCH3)',
  'UPPER PRIMARY ONLY (SCH4)',
  'UPPER PRIMARY WITH SEC./H.SEC (SCH5)',
  'PRIMARY WITH UPPER PRIMARY SEC (SCH6)',
  'UPPER PRIMARY WITH  SEC. (SCH7)',
  'NO RESPONSE (SCH9)',
]
plt.figure(figsize=(12, 6))
sns.violinplot(data=df[school_types])
plt.title('Distribution of Schools by Type')
plt.xticks(rotation=45)
plt.show()
plt.close()
```



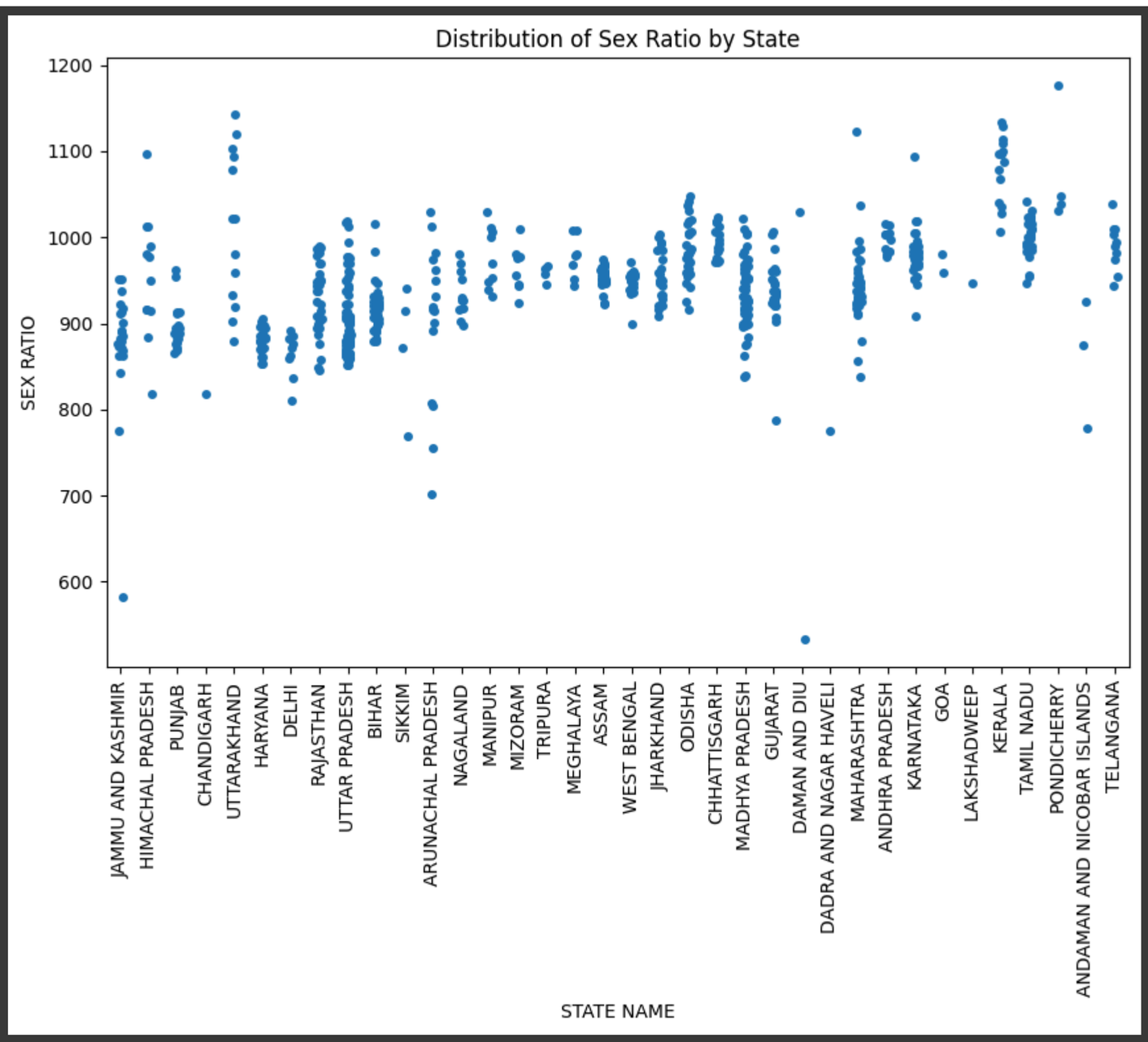

```
sns.regplot(x='TOTAL POULATION', y='TOTAL (ENRTOT)', data=df)
plt.title('Total Population vs Total Enrollment')
plt.show()
plt.close()
```



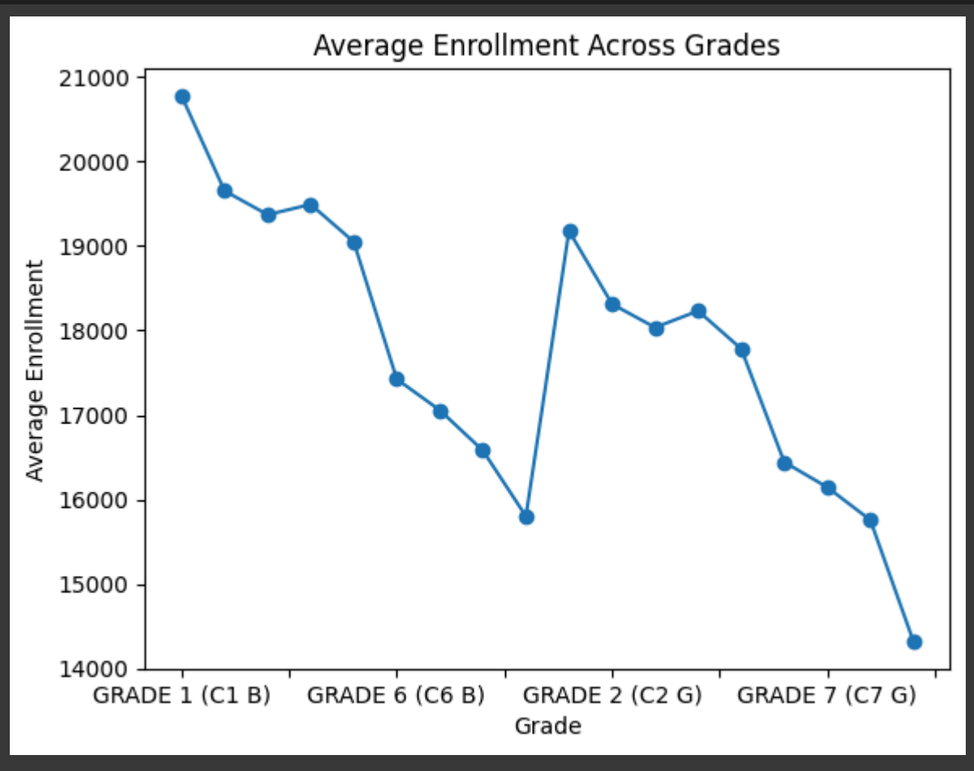

```
fig = px.scatter_3d(df, x='TOTAL POULATION', y='OVERALL LITERACY', z='TOTAL (ENRTOT)',
                    color='STATE NAME', title='Population, Literacy, and Enrollment')
fig.show()
```



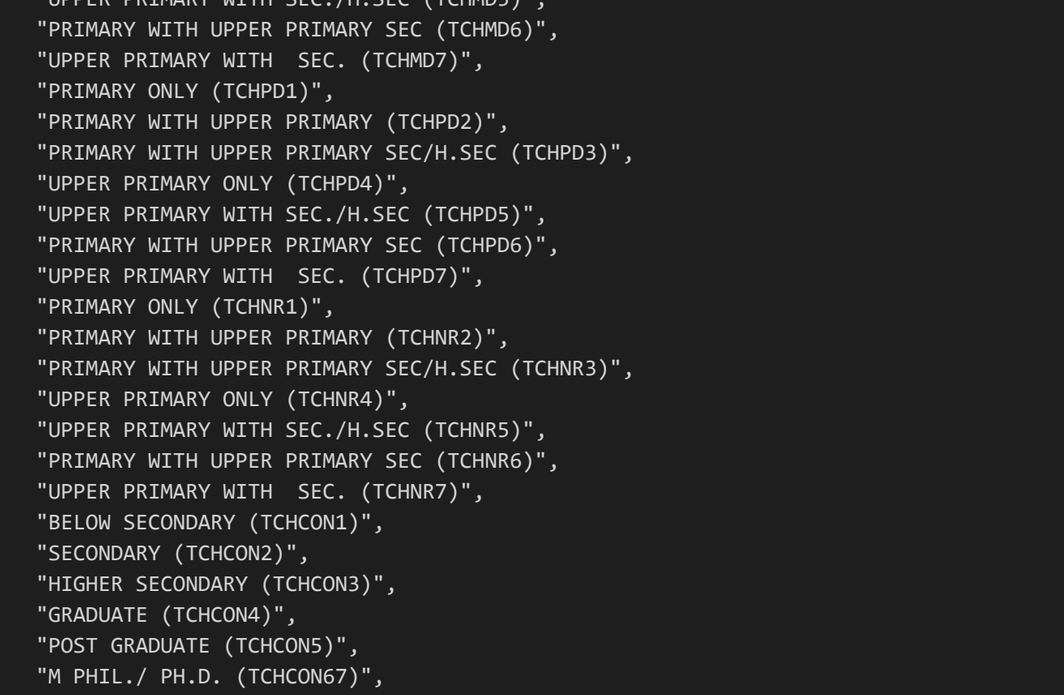

```
plt.figure(figsize=(10, 6))
sns.stripplot(x='STATE NAME', y='SEX RATIO', data=df, jitter=True)
plt.title('Distribution of Sex Ratio by State')
plt.xticks(rotation=90)
plt.show()
plt.close()
```



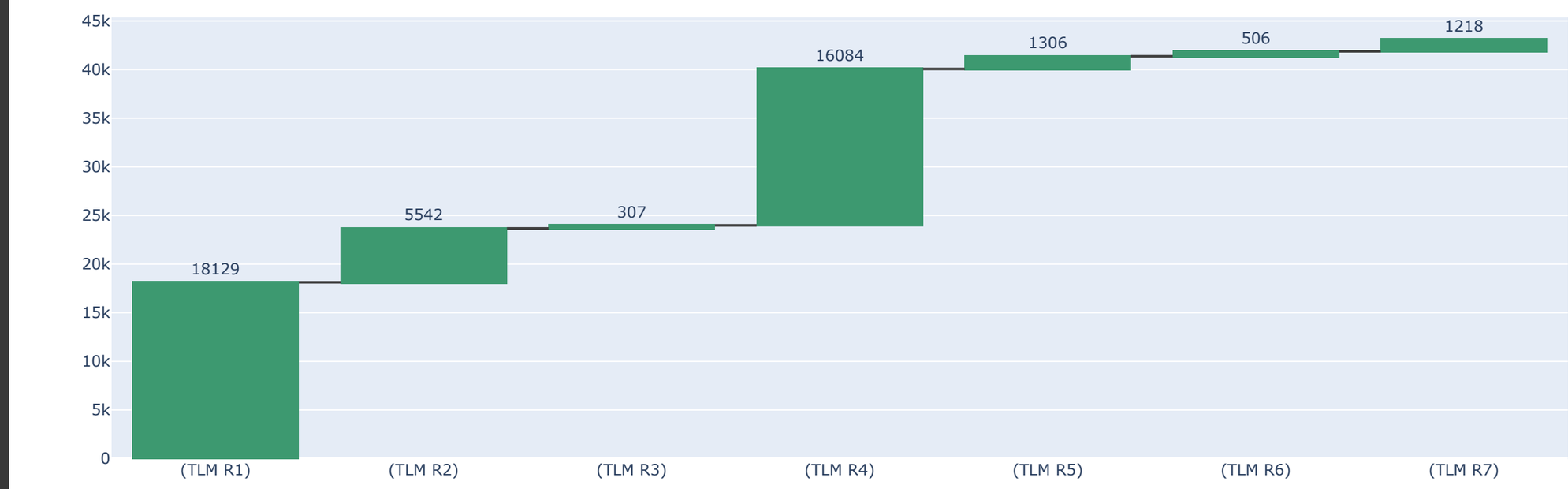

```
grade_columns = [
    "GRADE 1 (C1 B)",
    "GRADE 2 (C2 B)",
    "GRADE 3 (C3 B)",
    "GRADE 4 (C4 B)",
    "GRADE 5 (C5 B)",
    "GRADE 6 (C6 B)",
    "GRADE 7 (C7 B)",
    "GRADE 8 (C8 B)",
    "GRADE 9 (C9 B)",
    "GRADE 1 (C1 G)",
    "GRADE 2 (C2 G)",
    "GRADE 3 (C3 G)",
    "GRADE 4 (C4 G)",
    "GRADE 5 (C5 G)",
    "GRADE 6 (C6 G)",
    "GRADE 7 (C7 G)",
    "GRADE 8 (C8 G)",
    "GRADE 9 (C9 G)",
]
df[grade_columns].mean().plot(kind='line', marker='o')
plt.title('Average Enrollment Across Grades')
plt.xlabel('Grade')
plt.ylabel('Average Enrollment')
plt.show()
plt.close()
```



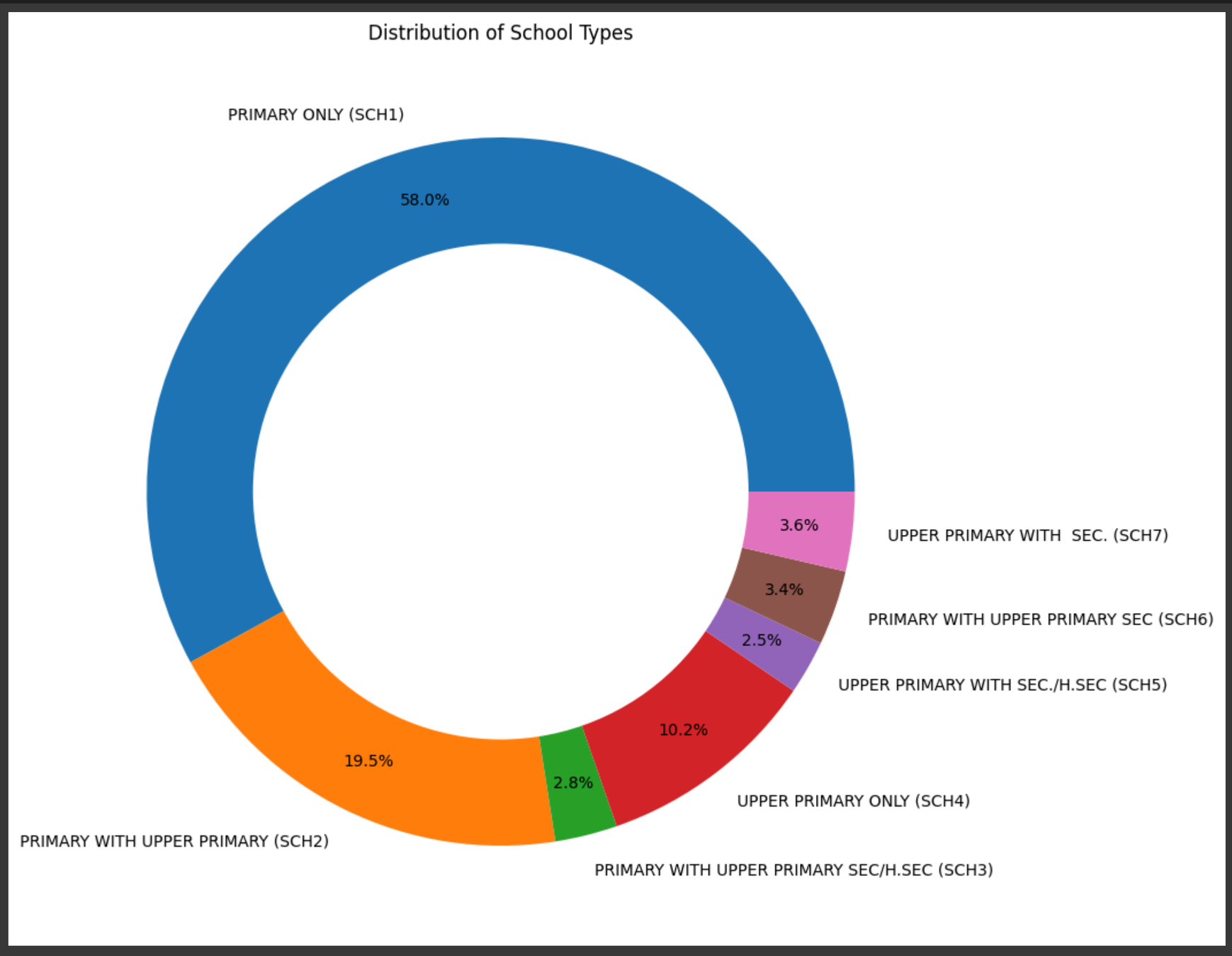
```
teacher_qual = [
    "PRIMARY ONLY (TCHBS1)",
    "PRIMARY WITH UPPER PRIMARY (TCHBS2)",
    "PRIMARY WITH UPPER PRIMARY SEC/H.SEC (TCHBS3)",
    "UPPER PRIMARY ONLY (TCHBS4)",
    "UPPER PRIMARY WITH SEC./H.SEC (TCHBS5)",
    "PRIMARY WITH UPPER PRIMARY SEC (TCHBS6)",
    "UPPER PRIMARY WITH  SEC. (TCHBS7)",
    "PRIMARY ONLY (TCHSEC1)",
    "PRIMARY WITH UPPER PRIMARY (TCHSEC2)",
    "PRIMARY WITH UPPER PRIMARY SEC/H.SEC (TCHSEC3)",
    "UPPER PRIMARY ONLY (TCHSEC4)",
    "UPPER PRIMARY WITH SEC./H.SEC (TCHSEC5)",
    "PRIMARY WITH UPPER PRIMARY SEC (TCHSEC6)",
    "UPPER PRIMARY WITH  SEC. (TCHSEC7)",
    "PRIMARY ONLY (TCHHS1)",
    "PRIMARY WITH UPPER PRIMARY (TCHHS2)",
    "PRIMARY WITH UPPER PRIMARY SEC/H.SEC (TCHHS3)",
    "UPPER PRIMARY ONLY (TCHHS4)",
    "UPPER PRIMARY WITH SEC./H.SEC (TCHHS5)",
    "PRIMARY WITH UPPER PRIMARY SEC (TCHHS6)",
    "UPPER PRIMARY WITH  SEC. (TCHHS7)",
    "PRIMARY ONLY (TCHGD1)",
    "PRIMARY WITH UPPER PRIMARY (TCHGD2)",
    "PRIMARY WITH UPPER PRIMARY SEC/H.SEC (TCHGD3)",
    "UPPER PRIMARY ONLY (TCHGD4)",
    "UPPER PRIMARY WITH SEC./H.SEC (TCHGD5)",
    "PRIMARY WITH UPPER PRIMARY SEC (TCHGD6)",
    "UPPER PRIMARY WITH  SEC. (TCHGD7)",
    "PRIMARY ONLY (TCHPG1)",
    "PRIMARY WITH UPPER PRIMARY (TCHPG2)",
    "PRIMARY WITH UPPER PRIMARY SEC/H.SEC (TCHPG3)",
    "UPPER PRIMARY ONLY (TCHPG4)",
    "UPPER PRIMARY WITH SEC./H.SEC (TCHPG5)",
    "PRIMARY WITH UPPER PRIMARY SEC (TCHPG6)",
    "UPPER PRIMARY WITH  SEC. (TCHPG7)",
    "PRIMARY ONLY (TCHMD1)",
    "PRIMARY WITH UPPER PRIMARY (TCHMD2)",
    "PRIMARY WITH UPPER PRIMARY SEC/H.SEC (TCHMD3)",
    "UPPER PRIMARY ONLY (TCHMD4)",
    "UPPER PRIMARY WITH SEC./H.SEC (TCHMD5)",
    "PRIMARY WITH UPPER PRIMARY SEC (TCHMD6)",
    "UPPER PRIMARY WITH  SEC. (TCHMD7)",
    "PRIMARY ONLY (TCHPD1)",
    "PRIMARY WITH UPPER PRIMARY (TCHPD2)",
    "PRIMARY WITH UPPER PRIMARY SEC/H.SEC (TCHPD3)",
    "UPPER PRIMARY ONLY (TCHPD4)",
    "UPPER PRIMARY WITH SEC./H.SEC (TCHPD5)",
    "PRIMARY WITH UPPER PRIMARY SEC (TCHPD6)",
    "UPPER PRIMARY WITH  SEC. (TCHPD7)",
    "PRIMARY ONLY (TCHNR1)",
    "PRIMARY WITH UPPER PRIMARY (TCHNR2)",
    "PRIMARY WITH UPPER PRIMARY SEC/H.SEC (TCHNR3)",
    "UPPER PRIMARY ONLY (TCHNR4)",
    "UPPER PRIMARY WITH SEC./H.SEC (TCHNR5)",
    "PRIMARY WITH UPPER PRIMARY SEC (TCHNR6)",
    "UPPER PRIMARY WITH  SEC. (TCHNR7)",
    "BELOW SECONDARY (TCHCON1)",
    "SECONDARY (TCHCON2)",
    "HIGHER SECONDARY (TCHCON3)",
    "GRADUATE (TCHCON4)",
    "POST GRADUATE (TCHCON5)",
    "M PHIL./ PH.D. (TCHCON67)",
  "POST DOCTORATE (TCHCON8)",

]
df[teacher_qual].mean().plot(kind='area')
plt.title('Distribution of Teachers by Qualification')
plt.xlabel('Qualification Level')
plt.ylabel('Average Number of Teachers')
plt.show()
plt.close()
```




```
import plotly.graph_objects as go
budget_columns = ['(TLM R1)', '(TLM R2)', '(TLM R3)', '(TLM R4)', '(TLM R5)', '(TLM R6)', '(TLM R7)']
budget_data = df[budget_columns].sum()

fig = go.Figure(go.Waterfall(
    name = "Budget Allocation",
    orientation = "v",
    measure = ["relative"] * len(budget_data),
    x = budget_data.index,
    textposition = "outside",
    text = budget_data.values,
    y = budget_data.values,
    connector = {"line":{"color":"rgb(63, 63, 63)"}},
))
fig.show()
```

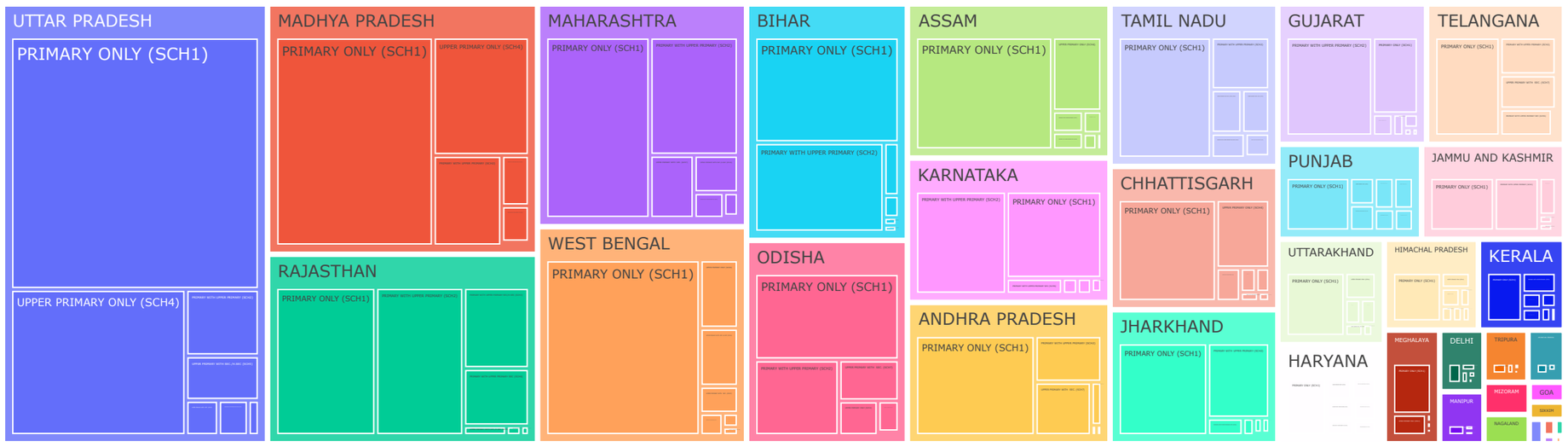



```
import matplotlib.pyplot as plt

school_types = [
  'PRIMARY ONLY (SCH1)',
  'PRIMARY WITH UPPER PRIMARY (SCH2)',
  'PRIMARY WITH UPPER PRIMARY SEC/H.SEC (SCH3)',
  'UPPER PRIMARY ONLY (SCH4)',
  'UPPER PRIMARY WITH SEC./H.SEC (SCH5)',
  'PRIMARY WITH UPPER PRIMARY SEC (SCH6)',
  'UPPER PRIMARY WITH  SEC. (SCH7)',
]
school_type_totals = df[school_types].sum()

fig, ax = plt.subplots(figsize=(10, 10))
ax.pie(school_type_totals, labels=school_types, autopct='%1.1f%%', pctdistance=0.85)

# Create a circle at the center to turn it into a donut chart
center_circle = plt.Circle((0,0), 0.60, fc='white')
fig.gca().add_artist(center_circle)

plt.title('Distribution of School Types')
plt.show()
```




```
df_grouped = df.groupby('STATE NAME')[school_types].sum().reset_index()
df_melted = pd.melt(df_grouped, id_vars=['STATE NAME'], value_vars=school_types,
                    var_name='School Type', value_name='Count')

fig = px.treemap(df_melted, path=['STATE NAME', 'School Type'], values='Count',
                 title='Enrollment by State and School Type')
fig.show()
```

Enrollment by State and School Type




```
enrollment_stages = [
    "PRIMARY ONLY (ENR1)",
    "PRIMARY WITH UPPER PRIMARY (ENR2)"
```