In [71]:
```
import os
import sys
import pandas as pd
import numpy as np
from scripts.parse_clear_admit import parse_clear_admit

input_filename = os.path.join(mba_folder_path, 'research', 'data', 'raw', 'clear_admit.txt')
output_filename = os.path.join(mba_folder_path, 'research', 'data', 'intermediate', 'clear_admit_parsed.csv')

notebook_dir = os.getcwd()
mba_folder_path = os.path.dirname(notebook_dir)
df = parse_clear_admit(input_filename, output_filename)
df.head(3)
```

Out[71]:

	School	GPA	GRE	Program Type	Application Location	Date	Round	Post-MBA Career	Note
0	Berkeley / Haas	3.1	356.0	NaN	NYC	2017-12-16	Round 1	Non Profit / Social Impact	Got a call on 12/11
1	Stanford GSB	3.6	354.0	NaN	Illinois	2019-05-15	Round 3		Deferred admissions. Call at around 1pm PST.
2	Berkeley / Haas	3.7	348.0	NaN	Boston, MA	2021-05-13	Rolling Admissions	Entrepreneurship	NaN

In [72]:
```
df = df[df['GPA'].between(0, 4)]
df = df[df['GRE'].between(0, 340)]
df = df[df['Date'].between('2022-01-01', '2025-01-01')]
df.to_csv('./data/cleaned/clear_admit_data.csv', index=False)
```

In [73]:
```
print(f"Total number of entries: {len(df)}")
print(f"Schools represented: {df['School'].nunique()}")
print(f"Average GRE score: {df['GRE'].mean():.2f}")
print(f"Average GPA: {df['GPA'].mean():.2f}")
```

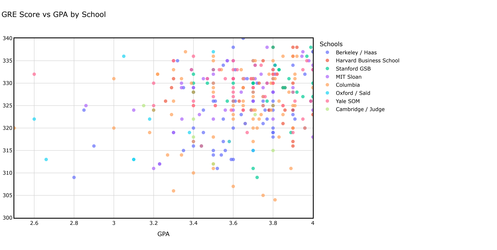
```
Total number of entries: 349
Schools represented: 8
Average GRE score: 326.07
Average GPA: 3.63
```

In [74]:
```
def calculate_quartiles(group):
    return pd.DataFrame({
        'GPA_Q1': round(group['GPA'].quantile(0.25), 1),
        'GPA_Q2': round(group['GPA'].quantile(0.5), 1),
        'GPA_Q3': round(group['GPA'].quantile(0.75), 1),
        'GRE_Q1': round(group['GRE'].quantile(0.25), 0).astype(int),
        'GRE_Q2': round(group['GRE'].quantile(0.5), 0).astype(int),
        'GRE_Q3': round(group['GRE'].quantile(0.75), 0).astype(int)
    }, index=[0])

school_quartiles = df.groupby('School').apply(calculate_quartiles, include_groups=False).reset_index().drop('level_1', axis=1)
school_quartiles_sorted = school_quartiles.sort_values('GPA_Q2', ascending=False)
school_quartiles_sorted
```

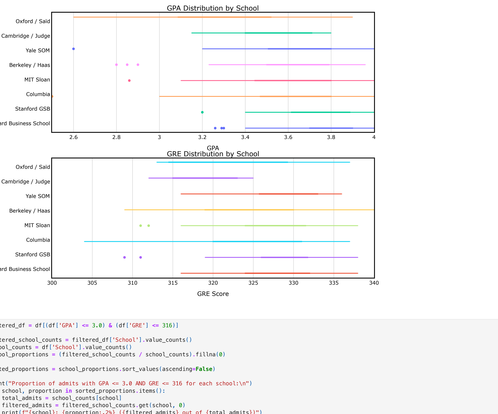
Out[74]:

	School	GPA_Q1	GPA_Q2	GPA_Q3	GRE_Q1	GRE_Q2	GRE_Q3
3	Harvard Business School	3.7	3.8	3.9	324	329	332
6	Stanford GSB	3.6	3.8	3.9	326	330	332
2	Columbia	3.5	3.7	3.8	320	326	331
4	MIT Sloan	3.4	3.7	3.8	324	329	331
0	Berkeley / Haas	3.5	3.6	3.8	319	325	330
7	Yale SOM	3.5	3.6	3.8	326	330	333
1	Cambridge / Judge	3.4	3.6	3.7	316	320	323
5	Oxford / Said	3.1	3.4	3.5	315	319	328

In [75]:
```
import plotly.graph_objects as go
import plotly.io as pio
from plotly.subplots import make_subplots

# Set the default renderer to 'notebook' for Jupyter
pio.renderers.default = "notebook"

# Create a figure
fig = go.Figure()

# Get unique schools
schools = df['School'].unique()

# Create a scatter plot for each school
for school in schools:
    school_data = df[df['School'] == school]
    fig.add_trace(go.Scatter(
        x=school_data['GPA'],
        y=school_data['GRE'],
        mode='markers',
        name=school,
        hovertemplate=
        '<b>%{text}</b><br>' +
        'GPA: %{x:.2f}<br>' +
        'GRE: %{y}<br>' +
        '<extra></extra>',
        text=[school] * len(school_data),
        marker=dict(size=8, opacity=0.7)
    ))

# Update layout with a light theme
fig.update_layout(
    title='GRE Score vs GPA by School',
    xaxis_title='GPA',
    yaxis_title='GRE Score',
    legend_title='Schools',
    hovermode='closest',
    width=1000,
    height=600,
    plot_bgcolor='white',
    paper_bgcolor='white',
    font=dict(color='black'),
    xaxis=dict(gridcolor='lightgrey', zerolinecolor='lightgrey'),
    yaxis=dict(gridcolor='lightgrey', zerolinecolor='lightgrey')
)

# Update axes
fig.update_xaxes(range=[2.5, 4.0], showline=True, linewidth=2, linecolor='black', mirror=True)
fig.update_yaxes(range=[300, 340], showline=True, linewidth=2, linecolor='black', mirror=True)

# Show the plot
fig.show()

# Optionally, save the plot as an interactive HTML file
fig.write_html("./data/visualizations/gre_vs_gpa_scatter_interactive.html")

config = {'scrollZoom': True, 'displayModeBar': True, 'responsive': True}
fig.write_html("./data/visualizations/gre_vs_gpa_scatter_interactive_light.html",
               config=config,
               include_plotlyjs='cdn',
               full_html=False)
```



In [76]:
```
import plotly.graph_objects as go
from plotly.subplots import make_subplots
import plotly.io as pio

# Set the default renderer to 'notebook' for Jupyter
pio.renderers.default = "notebook"

# Create subplots: one for GPA, one for GRE
fig = make_subplots(rows=2, cols=1,
                    subplot_titles=("GPA Distribution by School", "GRE Distribution by School"),
                    vertical_spacing=0.1)

# Get unique schools and sort them by median GPA (descending)
schools = df.groupby('School')['GPA'].median().sort_values(ascending=False).index

# Create box plots for GPA and GRE
for school in schools:
    school_data = df[df['School'] == school]

    # GPA box plot
    fig.add_trace(
        go.Box(x=school_data['GPA'], name=school, boxmean=True, orientation='h',
               hovertemplate="<b>%{x}</b><br>School: " + school + "<extra></extra>"),
        row=1, col=1
    )

    # GRE box plot
    fig.add_trace(
        go.Box(x=school_data['GRE'], name=school, boxmean=True, orientation='h',
               hovertemplate="<b>%{x}</b><br>School: " + school + "<extra></extra>"),
        row=2, col=1
    )

# Update layout
fig.update_layout(
    title='GPA and GRE Score Distributions by School',
    height=800,  # Increased height for better visibility
    width=1000,
    boxmode='group',
    plot_bgcolor='white',
    paper_bgcolor='white',
    font=dict(color='black'),
    showlegend=False
)

# Update x-axes
fig.update_xaxes(
    title_text="GPA",
    range=[2.5, 4.0],
    row=1, col=1,
    gridcolor='lightgrey',
    zerolinecolor='lightgrey',
    showline=True,
    linewidth=2,
    linecolor='black',
    mirror=True
)
fig.update_xaxes(
    title_text="GRE Score",
    range=[300, 340],
    row=2, col=1,
    gridcolor='lightgrey',
    zerolinecolor='lightgrey',
    showline=True,
    linewidth=2,
    linecolor='black',
    mirror=True
)

# Update y-axes
fig.update_yaxes(
    showline=True,
    linewidth=2,
    linecolor='black',
    mirror=True,
    automargin=True  # This helps to show full school names
)

# Show the plot
fig.show()

fig.write_html("./data/visualizations/gpa_gre_distributions_by_school.html")

config = {'scrollZoom': True, 'displayModeBar': True, 'responsive': True}
fig.write_html("./data/visualizations/gpa_gre_distributions_light.html",
               config=config,
               include_plotlyjs='cdn',
               full_html=False)
```



In [77]:
```
filtered_df = df[(df['GPA'] <= 3.0) & (df['GRE'] <= 316)]

filtered_school_counts = filtered_df['School'].value_counts()
school_counts = df['School'].value_counts()
school_proportions = (filtered_school_counts / school_counts).fillna(0)

sorted_proportions = school_proportions.sort_values(ascending=False)

print("Proportion of admits with GPA <= 3.0 AND GRE <= 316 for each school:\n")
for school, proportion in sorted_proportions.items():
    total_admits = school_counts[school]
    filtered_admits = filtered_school_counts.get(school, 0)
    print(f"{school}: {proportion:.2%} ({filtered_admits} out of {total_admits})")
```

```
Proportion of admits with GPA <= 3.0 AND GRE <= 316 for each school:

Oxford / Said: 7.69% (1 out of 13)
Berkeley / Haas: 2.63% (2 out of 76)
Cambridge / Judge: 0.00% (0 out of 6)
Columbia: 0.00% (0 out of 88)
Harvard Business School: 0.00% (0 out of 30)
MIT Sloan: 0.00% (0 out of 48)
Stanford GSB: 0.00% (0 out of 39)
Yale SOM: 0.00% (0 out of 49)
```

In [78]:
```
filtered_df = df[(df['GPA'] <= 3.0) | (df['GRE'] <= 316)]

filtered_school_counts = filtered_df['School'].value_counts()
school_counts = df['School'].value_counts()
school_proportions = (filtered_school_counts / school_counts).fillna(0)

sorted_proportions = school_proportions.sort_values(ascending=False)

print("Proportion of admits with GPA <= 3.0 OR GRE <= 316 for each school:\n")
for school, proportion in sorted_proportions.items():
    total_admits = school_counts[school]
    filtered_admits = filtered_school_counts.get(school, 0)
    print(f"{school}: {proportion:.2%} ({filtered_admits} out of {total_admits})")
```

```
Proportion of admits with GPA <= 3.0 OR GRE <= 316 for each school:

Oxford / Said: 46.15% (6 out of 13)
Cambridge / Judge: 33.33% (2 out of 6)
Berkeley / Haas: 18.42% (14 out of 76)
Columbia: 15.91% (14 out of 88)
MIT Sloan: 8.33% (4 out of 48)
Yale SOM: 6.12% (3 out of 49)
Stanford GSB: 5.13% (2 out of 39)
Harvard Business School: 3.33% (1 out of 30)
```