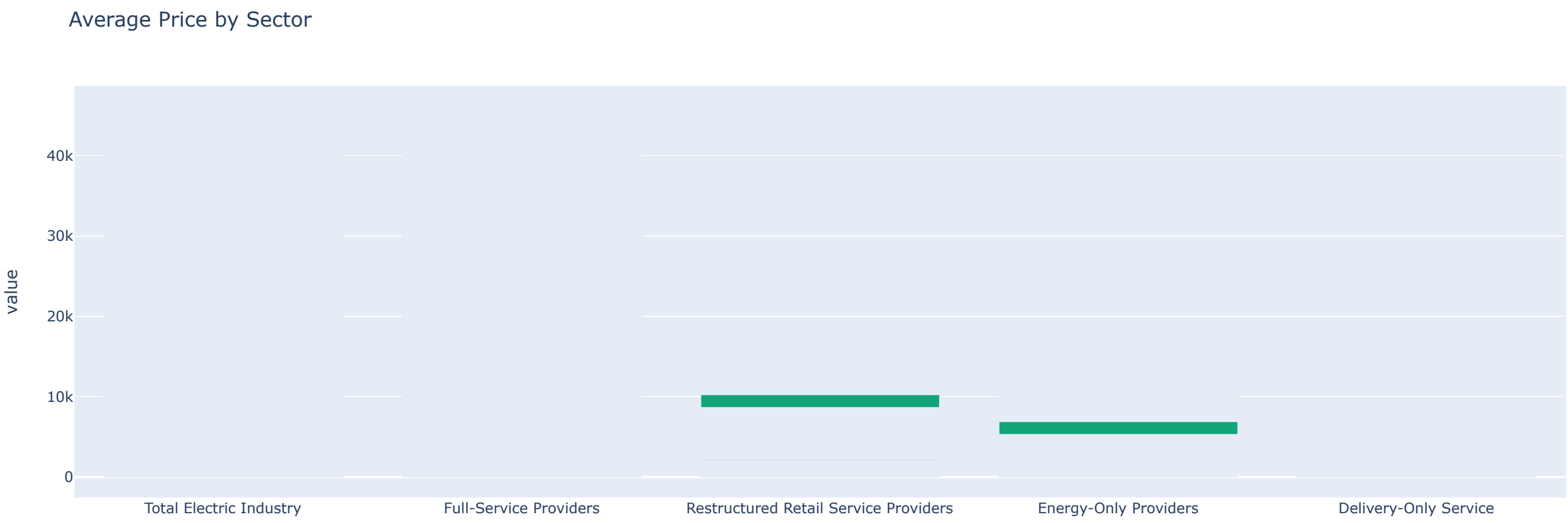
In [1]:
```
pip install dash pandas plotly
```

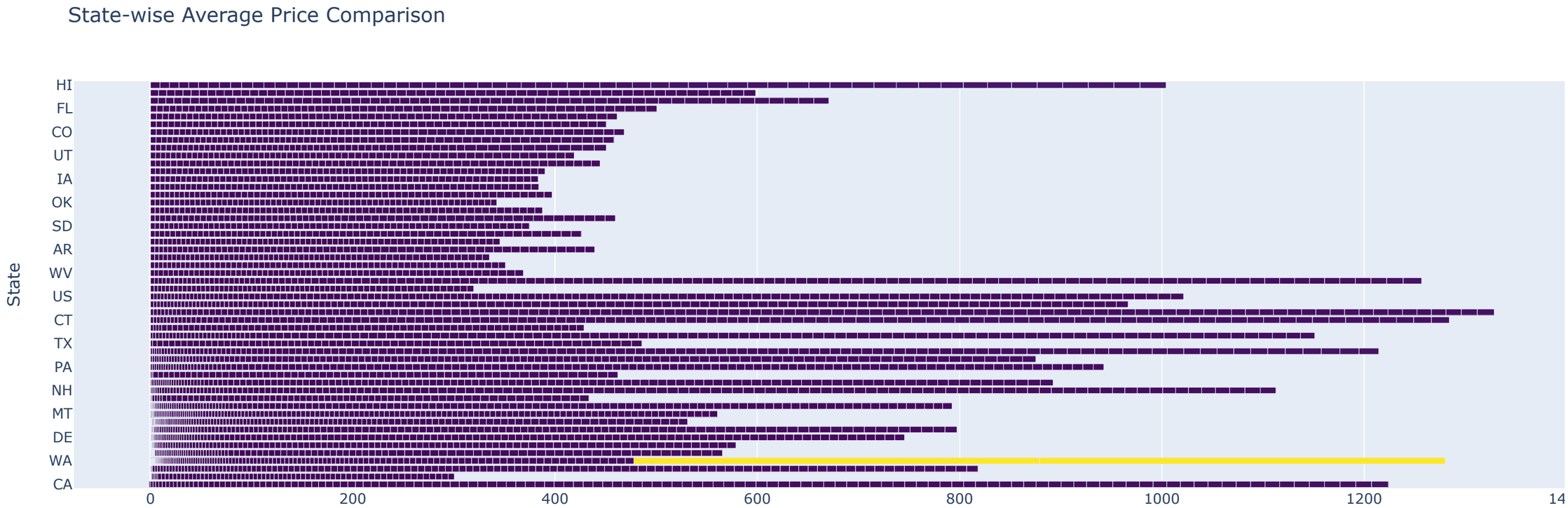
```
Requirement already satisfied: dash in /Users/rishabradesh/anaconda3/lib/python3.11/site-packages (2.16.1)
Requirement already satisfied: pandas in /Users/rishabradesh/anaconda3/lib/python3.11/site-packages (2.0.3)
Requirement already satisfied: plotly in /Users/rishabradesh/anaconda3/lib/python3.11/site-packages (5.9.0)
Requirement already satisfied: Flask<3.1,>=1.0.4 in /Users/rishabradesh/anaconda3/lib/python3.11/site-packages (from dash) (2.2.2)
Requirement already satisfied: Werkzeug<3.1 in /Users/rishabradesh/anaconda3/lib/python3.11/site-packages (from dash) (2.2.3)
Requirement already satisfied: dash-html-components==2.0.0 in /Users/rishabradesh/anaconda3/lib/python3.11/site-packages (from dash) (2.0.0)
Requirement already satisfied: dash-core-components==2.0.0 in /Users/rishabradesh/anaconda3/lib/python3.11/site-packages (from dash) (2.0.0)
Requirement already satisfied: dash-table==5.0.0 in /Users/rishabradesh/anaconda3/lib/python3.11/site-packages (from dash) (5.0.0)
Requirement already satisfied: importlib-metadata in /Users/rishabradesh/anaconda3/lib/python3.11/site-packages (from dash) (6.0.0)
Requirement already satisfied: typing-extensions>=4.1.1 in /Users/rishabradesh/anaconda3/lib/python3.11/site-packages (from dash) (4.7.1)
Requirement already satisfied: requests in /Users/rishabradesh/anaconda3/lib/python3.11/site-packages (from dash) (2.31.0)
Requirement already satisfied: retrying in /Users/rishabradesh/anaconda3/lib/python3.11/site-packages (from dash) (1.3.4)
Requirement already satisfied: nest-asyncio in /Users/rishabradesh/anaconda3/lib/python3.11/site-packages (from dash) (1.5.6)
Requirement already satisfied: setuptools in /Users/rishabradesh/anaconda3/lib/python3.11/site-packages (from dash) (68.0.0)
Requirement already satisfied: python-dateutil>=2.8.2 in /Users/rishabradesh/anaconda3/lib/python3.11/site-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /Users/rishabradesh/anaconda3/lib/python3.11/site-packages (from pandas) (2023.3.post1)
Requirement already satisfied: tzdata>=2022.1 in /Users/rishabradesh/anaconda3/lib/python3.11/site-packages (from pandas) (2023.3)
Requirement already satisfied: numpy>=1.21.0 in /Users/rishabradesh/anaconda3/lib/python3.11/site-packages (from pandas) (1.24.3)
Requirement already satisfied: tenacity>=6.2.0 in /Users/rishabradesh/anaconda3/lib/python3.11/site-packages (from plotly) (8.2.2)
Requirement already satisfied: Jinja2>=3.0 in /Users/rishabradesh/anaconda3/lib/python3.11/site-packages (from Flask<3.1,>=1.0.4->dash) (3.1.2)
Requirement already satisfied: itsdangerous>=2.0 in /Users/rishabradesh/anaconda3/lib/python3.11/site-packages (from Flask<3.1,>=1.0.4->dash) (2.0.1)
Requirement already satisfied: click>=8.0 in /Users/rishabradesh/anaconda3/lib/python3.11/site-packages (from Flask<3.1,>=1.0.4->dash) (8.0.4)
Requirement already satisfied: six>=1.5 in /Users/rishabradesh/anaconda3/lib/python3.11/site-packages (from python-dateutil>=2.8.2->pandas) (1.16.0)
Requirement already satisfied: MarkupSafe>=2.1.1 in /Users/rishabradesh/anaconda3/lib/python3.11/site-packages (from Werkzeug<3.1->dash) (2.1.1)
Requirement already satisfied: zipp>=0.5 in /Users/rishabradesh/anaconda3/lib/python3.11/site-packages (from importlib-metadata->dash) (3.11.0)
Requirement already satisfied: charset-normalizer<4,>=2 in /Users/rishabradesh/anaconda3/lib/python3.11/site-packages (from requests->dash) (2.0.4)
Requirement already satisfied: idna<4,>=2.5 in /Users/rishabradesh/anaconda3/lib/python3.11/site-packages (from requests->dash) (3.4)
Requirement already satisfied: urllib3<3,>=1.21.1 in /Users/rishabradesh/anaconda3/lib/python3.11/site-packages (from requests->dash) (1.26.16)
Requirement already satisfied: certifi>=2017.4.17 in /Users/rishabradesh/anaconda3/lib/python3.11/site-packages (from requests->dash) (2024.2.2)
Note: you may need to restart the kernel to use updated packages.
```

In [5]:
```
import pandas as pd

# Load the dataset
data_path = 'avgprice_annual.xlsx'
dataset = pd.read_excel(data_path)
```

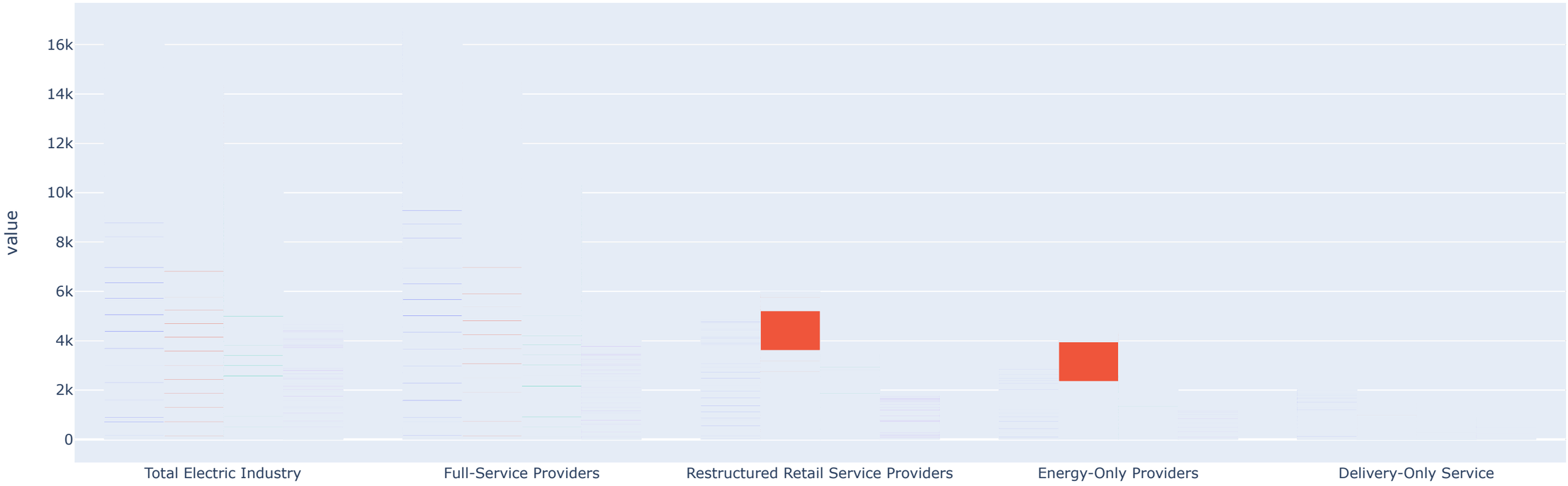
In [6]:
```
import plotly.express as px

#fig_sector_avg = px.bar(dataset, x="Industry Sector Category", y=["Residential", "Commercial", "Industrial", "Transportation"],
                        #title="Average Price by Sector")
fig_sector_avg = px.bar(
    dataset,
    x="Industry Sector Category",
    y=["Residential", "Commercial", "Industrial", "Transportation"],
    title="Average Price by Sector",
    # Set specific color for each sector for better contrast and readability
    color_discrete_sequence=px.colors.qualitative.Bold
)
fig_sector_avg.show()
```



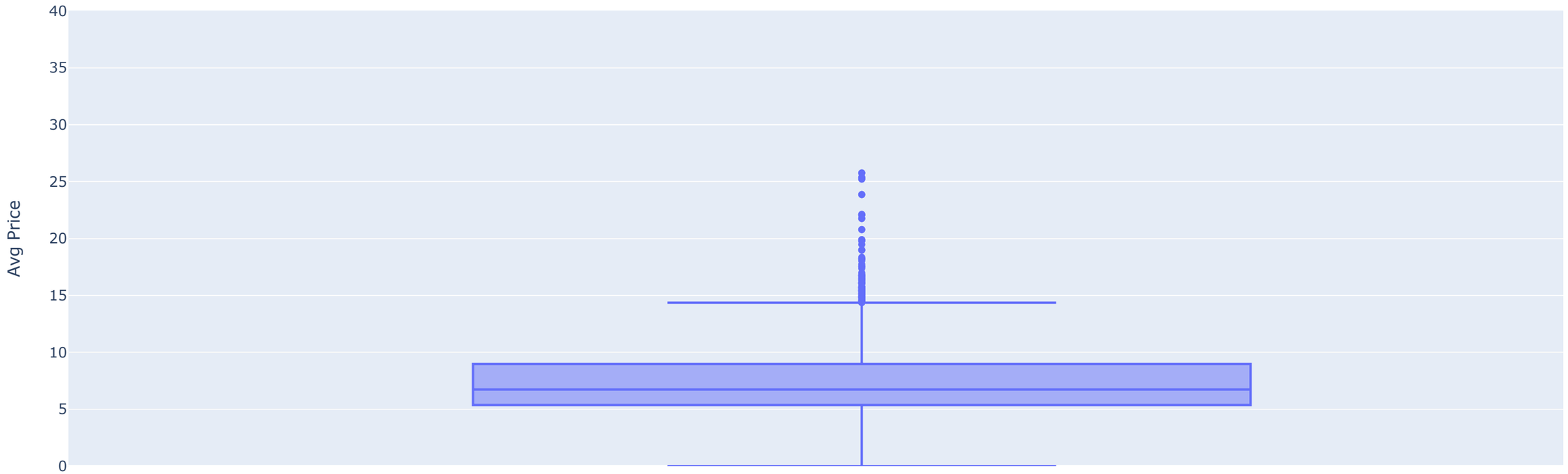
In [7]:
```
dataset['Average Price'] = dataset[["Residential", "Commercial", "Industrial", "Transportation"]].mean(axis=1)
#fig_state_comparison = px.bar(dataset.sort_values('Average Price'), x='Average Price', y='State', orientation='h',
                              #title="State-wise Average Price Comparison")

#fig_state_comparison.update_yaxes(range=[0, 40])

# Assuming 'dataset' is your DataFrame with the correct structure and 'Average Price' already calculated
fig_state_comparison = px.bar(
    dataset.sort_values('Average Price'),
    x='Average Price',
    y='State',
    orientation='h',
    title="State-wise Average Price Comparison",
    color='Average Price',  # This will color the bars based on the value of 'Average Price'
    color_continuous_scale=px.colors.sequential.Viridis  # This is an example of a more vivid color scale
)

# Show the figure
fig_state_comparison.show()
```



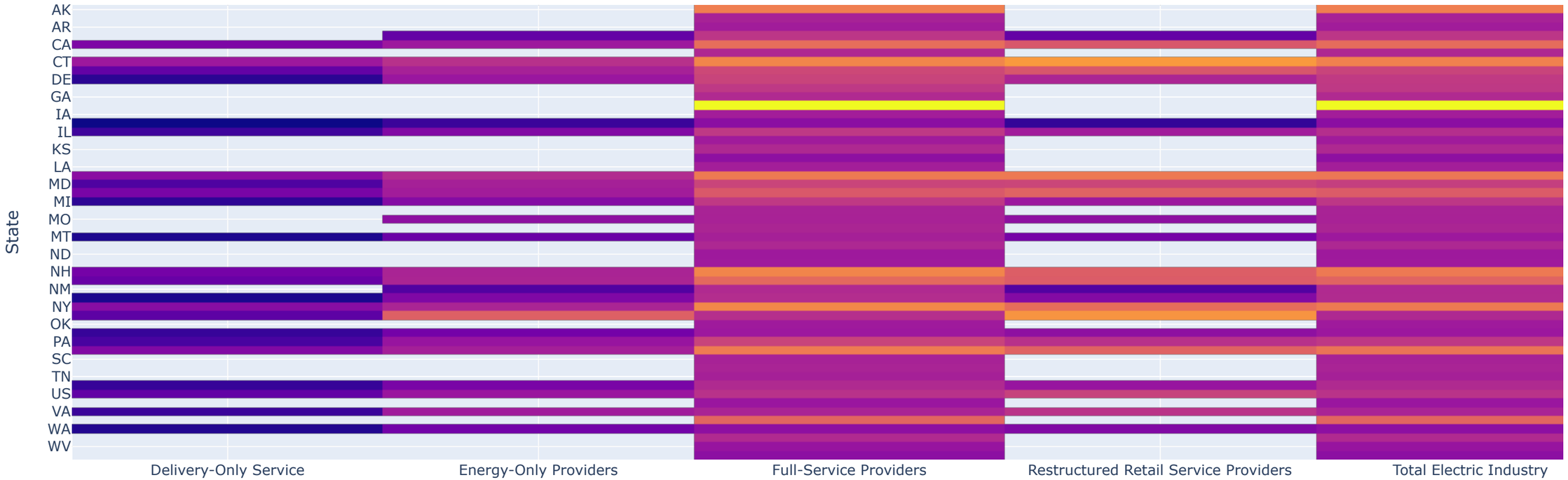
In [8]:
```
fig1 = px.bar(dataset, x='Industry Sector Category', y=['Residential', 'Commercial', 'Industrial', 'Transportation'], barmode='group')
fig1.show()
```



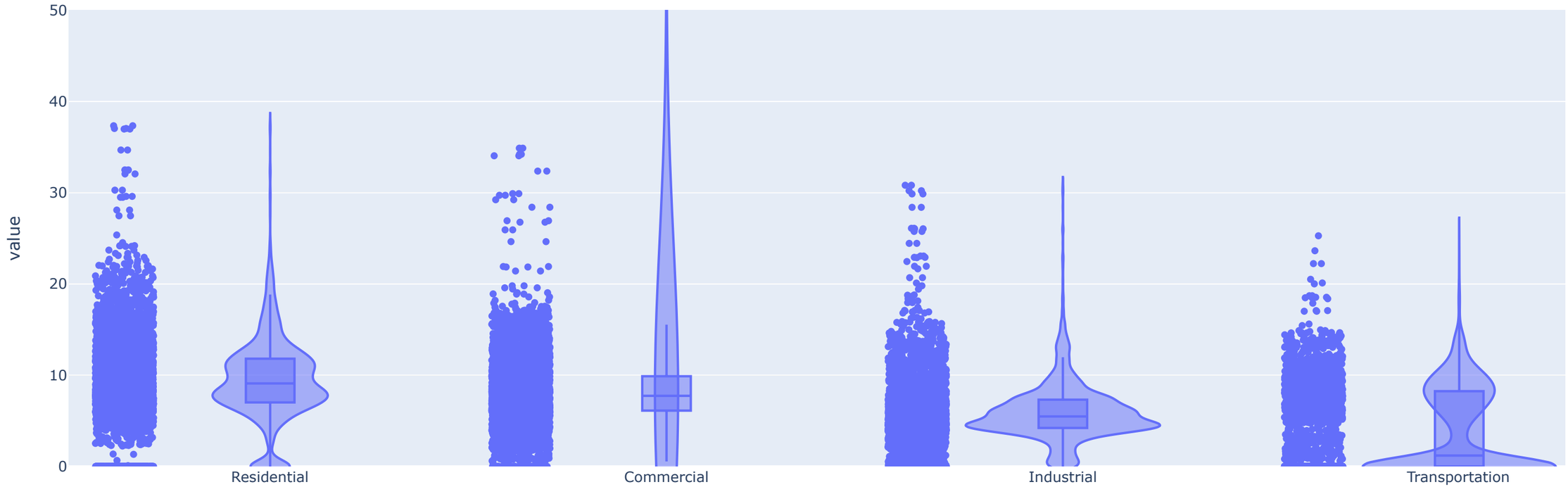
In [9]:
```
dataset['Average'] = dataset[['Residential', 'Commercial', 'Industrial', 'Transportation']].mean(axis=1)
fig2 = px.box(dataset, y='Average', labels={'Average': 'Avg Price'})
fig2.update_yaxes(range=[0, 40])

fig2.show()
```



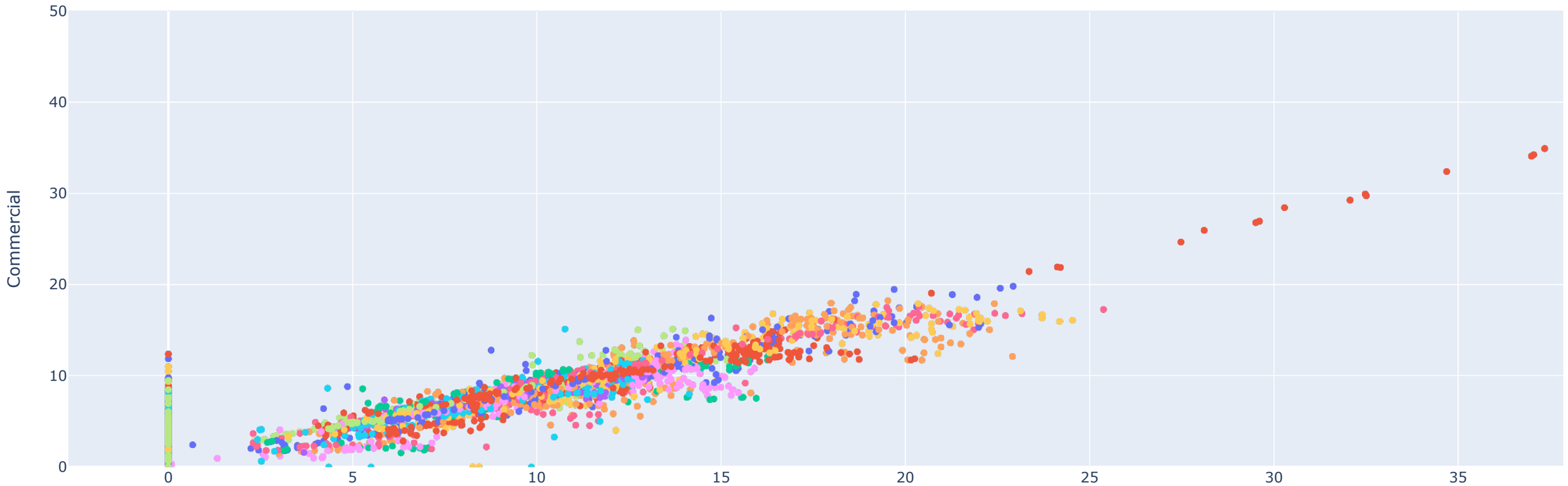
In [10]:
```
# Use pivot_table with aggregation to handle duplicate entries
heatmap_data = dataset.pivot_table(index='State', columns='Industry Sector Category', values='Total', aggfunc='mean')

# Now, plotting the heatmap
fig3 = px.imshow(heatmap_data, labels=dict(x="Sector", y="State", color="Price"), aspect="auto")

# Displaying the figure
fig3.show()
```



In [11]:
```
fig5 = px.violin(dataset, y=['Residential', 'Commercial', 'Industrial', 'Transportation'], box=True, points="all")
fig5.update_yaxes(range=[0, 50])
fig5.show()
```



In [12]:
```
fig6 = px.scatter(dataset, x='Residential', y='Commercial', color='State')
fig6.update_yaxes(range=[0, 50])
```



In [13]:
```
fig_violin = px.violin(dataset, y=["Residential", "Commercial", "Industrial", "Transportation"],
                       box=True, points="all", title="Sector Price Distribution")
```

In [14]:
```
from dash import Dash, html, dcc, Input, Output
import plotly.express as px  # Assuming you use Plotly Express for your figures

# Initialize the Dash app
app = Dash(__name__)

# App layout
app.layout = html.Div([
    html.H1("Electricity Price Dashboard", style={'text-align': 'center'}),
    dcc.Dropdown(id='state-dropdown',
                 options=[{'label': state, 'value': state} for state in dataset['State'].unique()],
                 value=dataset['State'].unique()[0], style={'width': '50%', 'margin': '0 auto'}),
    # First graph in its own row, styled for better appearance
    html.Div([dcc.Graph(id='sector-price-line-chart')], style={'width': '100%'}),
    # Subsequent rows with two graphs each, styled for better appearance
    html.Div([
        html.Div([dcc.Graph(figure=fig_sector_avg)], style={'display': 'inline-block', 'width': '50%'}),
        html.Div([dcc.Graph(figure=fig_state_comparison)], style={'display': 'inline-block', 'width': '50%'})
    ]),
    html.Div([
        html.Div([dcc.Graph(figure=fig6)], style={'display': 'inline-block', 'width': '50%'}),
        html.Div([dcc.Graph(figure=fig3)], style={'display': 'inline-block', 'width': '50%'})
    ]),
    html.Div([
        html.Div([dcc.Graph(figure=fig2)], style={'display': 'inline-block', 'width': '50%'}),
        html.Div([dcc.Graph(figure=fig5)], style={'display': 'inline-block', 'width': '50%'})
    ]),


])

# Callback to update the line chart with enhanced appearance
@app.callback(
    Output('sector-price-line-chart', 'figure'),
    Input('state-dropdown', 'value')
)
def update_line_chart(selected_state):
    filtered_data = dataset[dataset['State'] == selected_state]
    fig = px.line(filtered_data, x="Year", y=["Residential", "Commercial", "Industrial", "Transportation"],
                  title=f"Price Trends by Sector for {selected_state}")
    # Improve graph aesthetics
    fig.update_layout(
        template='plotly_white',
        autosize=True,
        margin=dict(l=40, r=40, t=40, b=40),
        font=dict(size=12),
    )
    return fig

# Run the app
if __name__ == '__main__':
    app.run_server(debug=True, port=8057, jupyter_mode="external")  # Use any available port
```

```
Dash app running on http://127.0.0.1:8057/
```

In [30]:
```
dataset
```

Out[30]:

	Year	State	Industry Sector Category	Residential	Commercial	Industrial	Transportation	Other	Total	Average Price	Average
0	2020	AK	Total Electric Industry	22.57	19.58	15.88	0.00	NaN	19.82	14.507500	14.507500
1	2020	AL	Total Electric Industry	12.58	11.55	5.87	0.00	NaN	9.84	7.500000	7.500000
2	2020	AR	Total Electric Industry	10.41	8.61	5.89	13.32	NaN	8.32	9.557500	9.557500
3	2020	AZ	Total Electric Industry	12.27	10.11	6.07	9.38	NaN	10.44	9.457500	9.457500
4	2020	CA	Total Electric Industry	20.45	17.53	14.27	10.07	NaN	18.00	15.580000	15.580000
...	...	...	...	...	...	...	...	...	...	...	...
4600	1990	WA	Full-Service Providers	4.39	4.15	2.39	NaN	3.13	3.40	3.643333	3.643333
4601	1990	WI	Full-Service Providers	6.63	5.78	3.99	NaN	6.47	5.37	5.466667	5.466667
4602	1990	WV	Full-Service Providers	5.90	5.36	3.56	NaN	8.19	4.73	4.940000	4.940000
4603	1990	WY	Full-Service Providers	5.97	5.17	3.47	NaN	7.90	4.21	4.870000	4.870000
4604	1990	US	Full-Service Providers	7.83	7.34	4.74	NaN	6.40	6.57	6.636667	6.636667

4605 rows × 11 columns

In [ ]:
```

```

In [ ]:
```

```