Report: Analysis of India's Energy Landscape

Introduction

Hello! I'm Jayalakshmi, and in this report, I present an analysis of India's energy landscape based on comprehensive yearly data provided by Ember. This dataset encompasses electricity generation, emissions, and carbon intensity metrics from 2010 to 2023, offering insights into the country's energy sector dynamics and environmental impact.

Data Source

The primary dataset used for this analysis is sourced from Ember, providing detailed yearly records on:

Electricity Generation (in GWh) Power Sector Emissions (in ktCO2) Carbon Intensity (in gCO2/kWh) The dataset captures these metrics across various sectors and technologies, enabling a comprehensive examination of trends and correlations over time.

Methodology and Tools

For this analysis, Python programming language was utilized along with the Plotly library for interactive data visualization. Below are example visualizations created from the Ember dataset to illustrate key insights:

```
#Importing Library
import pandas as pd
import plotly.express as px
import plotly.graph_objects as go
from plotly.subplots import make_subplots
```

EDA

```
india_electricity_df = pd.read_csv('/content/sample_data/india_yearly_full_release_long_format-5.csv')
```

```
india_electricity_df.head()
```

	Country	Country code	State	State code	State type	Year	Category	Subcategory	Variable	Unit	Value	YoY absolute change
0	India	IND	Andaman and Nicobar	AN	Union territory	2019	Electricity generation	Aggregate fuel	Clean	%	19.69	NaN
1	India	IND	Andaman and Nicobar	AN	Union territory	2019	Electricity generation	Aggregate fuel	Fossil	%	80.31	NaN

```
india_electricity_df.shape
```

```
(11622, 13)
```

```
india_electricity_df.columns
```

```
Index(['Country', 'Country code', 'State', 'State code', 'State type', 'Year',
       'Category', 'Subcategory', 'Variable', 'Unit', 'Value',
       'YoY absolute change', 'YoY % change'],
      dtype='object')
```

```
india_electricity_df['Category'].unique()
```

```
array(['Electricity generation', 'Power sector emissions'], dtype=object)
```

```
india_electricity_df['Unit'].unique()
```

```
array(['%', 'GWh', 'ktCO2', 'gCO2/kWh'], dtype=object)
```

```
india_electricity_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11622 entries, 0 to 11621
Data columns (total 13 columns):
 #   Column               Non-Null Count  Dtype  
---  ------               --------------  -----  
 0   Country              11622 non-null  object 
 1   Country code         11622 non-null  object 
 2   State                11622 non-null  object 
 3   State code           11622 non-null  object 
 4   State type           11622 non-null  object 
 5   Year                 11622 non-null  int64  
 6   Category             11622 non-null  object 
 7   Subcategory          11622 non-null  object 
 8   Variable             11622 non-null  object 
 9   Unit                 11622 non-null  object 
 10  Value                11574 non-null  float64
 11  YoY absolute change  4834 non-null   float64
 12  YoY % change         3003 non-null   float64
dtypes: float64(3), int64(1), object(9)
memory usage: 1.2+ MB
```

Removing % unit columns

```
# Filter out rows where the unit is '%'
filtered_data = india_electricity_df[india_electricity_df['Unit'] != '%']

# Verify the filtering
print(filtered_data['Unit'].unique())
```

```
['GWh' 'ktCO2' 'gCO2/kWh']
```

Annual Trends in Electricity Generation in INDIA

```
# Filter data for India only
india_data = filtered_data[filtered_data['State'] == 'India Total']
```

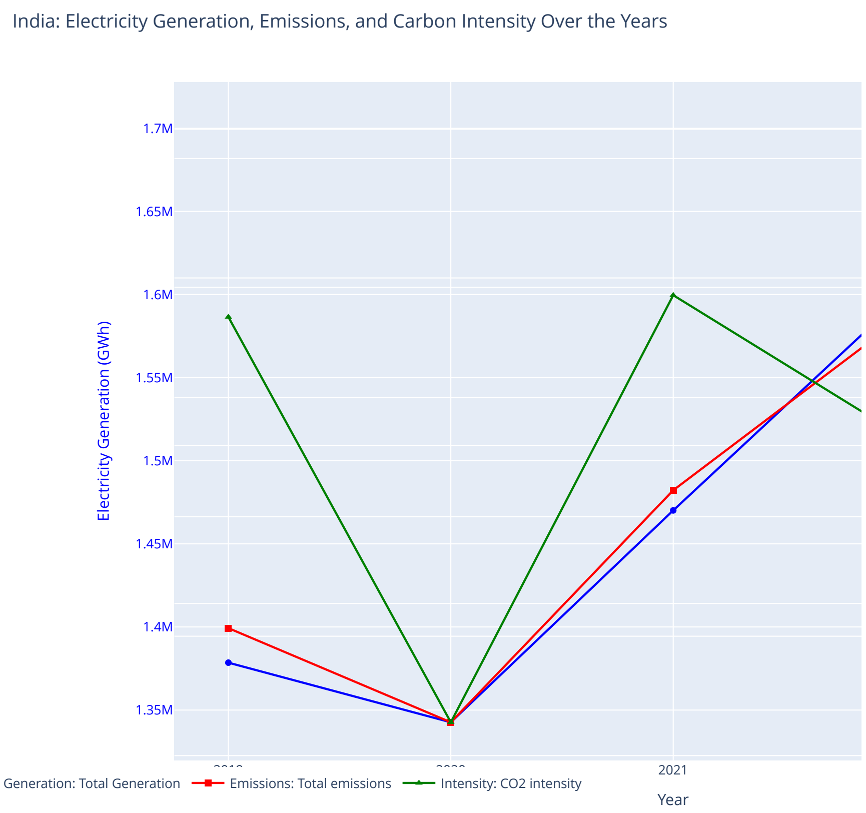
```
# Electricity Generation (GWh) Over the Years
india_generation = india_data[(india_data['Category'] == 'Electricity generation') &
                              (india_data['Unit'] == 'GWh') &
                              (india_data['Subcategory'] == 'Total')]

# Emissions (ktCO2) Over the Years
india_emissions = india_data[(india_data['Category'] == 'Power sector emissions') &
                             (india_data['Unit'] == 'ktCO2')&
                              (india_data['Subcategory'] == 'Total')]

# Carbon Intensity (gCO2/kWh) Over the Years
india_intensity = india_data[(india_data['Subcategory'] == 'CO2 intensity') &
                             (india_data['Unit'] == 'gCO2/kWh')]
```

```
india_data['Category'].unique()
```

```
array(['Electricity generation', 'Power sector emissions'], dtype=object)
```

```
# Create subplots
fig = make_subplots(rows=3, cols=1, shared_xaxes=True,
                    subplot_titles=('Electricity Generation (GWh)',
                                    'Power Sector Emissions (ktCO2)',
                                    'Carbon Intensity (gCO2/kWh)'))
```

```
# Create figure
fig = go.Figure()

# Add electricity generation traces
for variable in india_generation['Variable'].unique():
    subset = india_generation[india_generation['Variable'] == variable]
    fig.add_trace(go.Scatter(x=subset['Year'], y=subset['Value'], mode='lines+markers',
                             name=f'Generation: {variable}', yaxis='y1',
                             marker=dict(symbol='circle'), line=dict(color='blue')))

# Add emissions traces
for variable in india_emissions['Variable'].unique():
    subset = india_emissions[india_emissions['Variable'] == variable]
    fig.add_trace(go.Scatter(x=subset['Year'], y=subset['Value'], mode='lines+markers',
                             name=f'Emissions: {variable}', yaxis='y2',
                             marker=dict(symbol='square'), line=dict(color='red')))

# Add carbon intensity traces
for variable in india_intensity['Variable'].unique():
    subset = india_intensity[india_intensity['Variable'] == variable]
    fig.add_trace(go.Scatter(x=subset['Year'], y=subset['Value'], mode='lines+markers',
                             name=f'Intensity: {variable}', yaxis='y3',
                             marker=dict(symbol='triangle-up'), line=dict(color='green')))

# Update layout
fig.update_layout(
    title='India: Electricity Generation, Emissions, and Carbon Intensity Over the Years',
    xaxis=dict(title='Year', type='category'),
    yaxis=dict(
        title='Electricity Generation (GWh)',
        titlefont=dict(color='blue'),
        tickfont=dict(color='blue')
    ),
    yaxis2=dict(
        title='Emissions (ktCO2)',
        titlefont=dict(color='red'),
        tickfont=dict(color='red'),
        anchor='x',
        overlaying='y',
        side='right'
    ),
    yaxis3=dict(
        title='Carbon Intensity (gCO2/kWh)',
        titlefont=dict(color='green'),
        tickfont=dict(color='green'),
        anchor='free',
        overlaying='y',
        side='right',
        position=1
    ),
    legend=dict(x=0.1, y=-0.01, xanchor='center', orientation='h'),
    height=800,  # Increase the height of the plot
    width=1200   # Increase the width of the plot
)

# Show plot
fig.show()
```



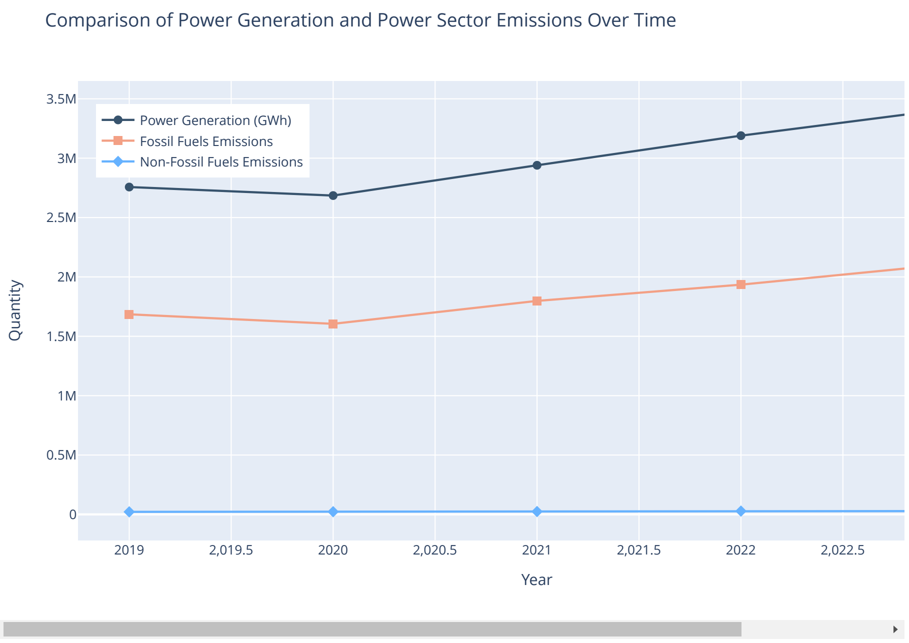
```
# Observation report
observation_report = """
**Observations:**
1. **Electricity Generation**: There is a steady increase in the total electricity generation (GWh) in India from 2019 to 2023.
2. **Emissions**: The total emissions (ktCO2) from electricity generation show a slight upward trend over the years, indicating that while generation is increasing, emissic
3. **Carbon Intensity**: The carbon intensity (gCO2/kWh) remains relatively constant over the observed period, suggesting that improvements in efficiency or shifts to clear
"""

print(observation_report)
```

```
**Observations:**
1. **Electricity Generation**: There is a steady increase in the total electricity generation (GWh) in India from 2019 to 2023.
2. **Emissions**: The total emissions (ktCO2) from electricity generation show a slight upward trend over the years, indicating that while generation is increasing, emi
3. **Carbon Intensity**: The carbon intensity (gCO2/kWh) remains relatively constant over the observed period, suggesting that improvements in efficiency or shifts to c
```

Power sector emissions fossil fuels and non fossil fuels

```
# Filter for 'Electricity generation' category
df_generation = filtered_data[(filtered_data['Category'] == 'Electricity generation') &
                              (filtered_data['Unit'] == 'GWh') &
                              (filtered_data['Subcategory'] == 'Total')]

# Filter for 'Power sector emissions' category
df_emissions = filtered_data[filtered_data['Category'] == 'Power sector emissions']

# List of fossil fuel and non-fossil fuel categories
fossil_fuels = ['Coal', 'Gas', 'Oil']  # You may adjust this based on your specific dataset
non_fossil_fuels = ['Hydro', 'Nuclear', 'Wind', 'Solar', 'Bioenergy', 'Other Renewables']

# Filter the DataFrame for fossil fuels and non-fossil fuels emissions
df_fossil_fuels_emissions = df_emissions[df_emissions['Variable'].isin(fossil_fuels)]
df_non_fossil_fuels_emissions = df_emissions[df_emissions['Variable'].isin(non_fossil_fuels)]

# Calculate total emissions for each category
total_fossil_emissions = df_fossil_fuels_emissions.groupby('Year')['Value'].sum()
total_non_fossil_emissions = df_non_fossil_fuels_emissions.groupby('Year')['Value'].sum()

# Calculate total generation for each year
total_generation = df_generation.groupby('Year')['Value'].sum()
```

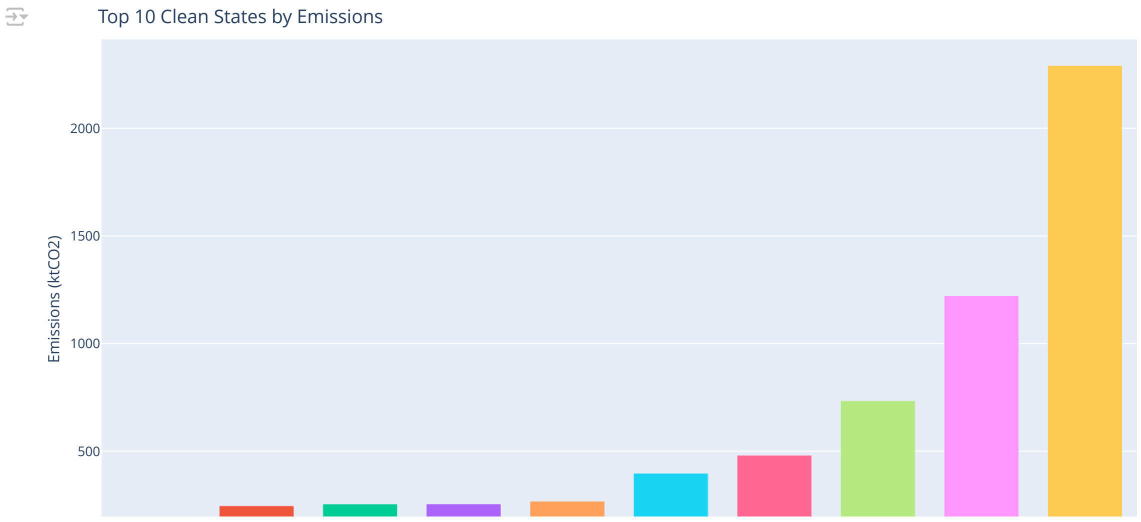
```
# Create a line chart to compare generation and emissions over time
fig = go.Figure()

# Add line traces for power generation, fossil fuels emissions, and non-fossil fuels emissions
fig.add_trace(go.Scatter(x=total_generation.index, y=total_generation.values,
                         mode='lines+markers', name='Power Generation (GWh)',
                         line=dict(color='rgb(55, 83, 109)', width=2),
                         marker=dict(symbol='circle', size=8)
                         ))

fig.add_trace(go.Scatter(x=total_fossil_emissions.index, y=total_fossil_emissions.values,
                         mode='lines+markers', name='Fossil Fuels Emissions',
                         line=dict(color='rgb(243, 160, 133)', width=2),
                         marker=dict(symbol='square', size=8)
                         ))

fig.add_trace(go.Scatter(x=total_non_fossil_emissions.index, y=total_non_fossil_emissions.values,
                         mode='lines+markers', name='Non-Fossil Fuels Emissions',
                         line=dict(color='rgb(102, 178, 255)', width=2),
                         marker=dict(symbol='diamond', size=8)
                         ))

# Update layout
fig.update_layout(
    title='Comparison of Power Generation and Power Sector Emissions Over Time',
    xaxis_title='Year',
    yaxis_title='Quantity',
    legend=dict(x=0.02, y=0.95),
    height=600,  # Adjust height as needed
    width=1000,  # Adjust width as needed
    hovermode='x unified'  # Show hover information for all lines at once
)

fig.show()
```



Top 10 Clean States and Last 10 Clean States

```
# Example: Filter data for emissions
emissions_data = filtered_data[filtered_data['Category'] == 'Power sector emissions']

# Aggregate data by state and calculate total emissions
state_emissions = emissions_data.groupby('State')['Value'].sum().reset_index()

# Rank states based on emissions (assuming lower emissions indicate cleaner states)
state_emissions_sorted = state_emissions.sort_values(by='Value',ascending =True )

# Select top 10 clean states and last 10 clean states
top_10_clean_states = state_emissions_sorted[state_emissions_sorted['State'] != 'Others'].head(10)
last_10_clean_states = state_emissions_sorted[state_emissions_sorted['State'] != 'India Total'].tail(10)
```

```
def plot_top_last_clean_states(states_data, title,ascending=True):
    """
    Function to plot top or last clean states based on emissions data.

    Parameters:
    - states_data: DataFrame containing 'State' and 'Value' columns.
    - title: Title of the plot.
    """
# True if plotting top states (ascending order), False if plotting last states (descending order)
    states_data_sorted = states_data.sort_values(by='Value', ascending= ascending)
    fig = px.bar(states_data, x='State', y='Value',
                 title=title,
                 labels={'State': 'State', 'Value': 'Emissions (ktCO2)'},
                 hover_name='State',
                 color='State',  # Optional: Color by state for differentiation
                 height=600      # Adjust height as needed
                )
# Update layout
    fig.update_layout(
        xaxis_title='',  # Remove x-axis title
        xaxis_showticklabels=False,
        yaxis_title='Emissions (ktCO2)',
        margin=dict(l=0, r=0, t=30, b=0),
        xaxis_tickangle=-45,
        bargap=0.2,    # Adjust gap between bars
        bargroupgap=0.1,  # Adjust gap between bar groups
        width=1000,    # Adjust width as needed
        legend=dict(
            orientation='h',  # Horizontal orientation for legend
            yanchor='bottom',  # Anchor legend to the bottom
            y=-0.2            # Position of legend below the plot (adjust as needed)
        )
    )

    fig.show()

# Call the function for top 10 clean states
plot_top_last_clean_states(top_10_clean_states, 'Top 10 Clean States by Emissions', ascending=True)
```


Top 10 Clean States by Emissions
Emissions (ktCO2)