```
import pandas as pd
import geopandas as gpd
import matplotlib.pyplot as plt
import os

# --- Configuration ---
DATA_FILE = 'neonatal_datascience.csv'
SHAPEFILE_PATH = 'Africa_Boundaries.shp' # <<< Check this path and
ensure the .shx file is present!

EAC_COUNTRIES = ['KEN', 'UGA', 'TZA', 'SOM', 'SSD', 'RWA', 'BDI',
'COD'] # ISO 3166-1 alpha-3 codes
EAC_COUNTRY_NAMES = { # Mapping for filtering/display
    'KEN': 'Kenya',
    'UGA': 'Uganda',
    'TZA': 'United Republic of Tanzania',
    'SOM': 'Somalia',
    'SSD': 'South Sudan',
    'RWA': 'Rwanda',
    'BDI': 'Burundi',
    'COD': 'Democratic Republic of The Congo'
}

# --- Load and Filter Data ---
try:
    # Add low_memory=False to potentially handle DtypeWarning better,
though it might not fix it
    df = pd.read_csv(DATA_FILE, low_memory=False)
    print(f"Successfully loaded {DATA_FILE}")
except FileNotFoundError:
    print(f"Error: {DATA_FILE} not found. Please ensure the CSV file
is in the correct directory.")
    exit()
except Exception as e:
    print(f"Error loading CSV file: {e}")
    exit()

# Check if required columns exist
# Corrected: Changed 'OBS_VALUE' and 'CONNECTION' to 'Observation
Value' and 'Observation Status'
required_cols = ['REF_AREA', 'Indicator', 'Sex', 'Wealth Quintile',
'Reference Date', 'Observation Value', 'Observation Status',
'Geographic area']
if not all(col in df.columns for col in required_cols):
    missing = [col for col in required_cols if col not in df.columns]
    print(f"Error: Missing required columns in CSV: {missing}")
    print("Available columns:", df.columns.tolist())
    exit()
```

```
# Filter data for EAC countries, 'Total' Sex/Wealth, and included in
IGME
# Corrected: Changed 'OBS_VALUE' and 'CONNECTION' references
df_filtered = df[
    (df['REF_AREA'].isin(EAC_COUNTRIES)) &
    (df['Sex'] == 'Total') &
    (df['Wealth Quintile'] == 'Total') &
    (df['Observation Status'] == 'Included in IGME')
].copy()


if df_filtered.empty:
    print("No data found for the specified EAC countries, Sex='Total',
Wealth Quintile='Total', and Observation Status='Included in IGME'.")
    print("Please check your CSV data and filtering criteria.")
    exit()

print(f"Filtered data includes {len(df_filtered)} rows for EAC
countries, Total Sex/Wealth, Included in IGME.")

# Convert 'Reference Date' to a numeric year (float like YYYY.Y seems
appropriate for plotting trends)
try:
    df_filtered['Year'] = pd.to_numeric(df_filtered['Reference Date'],
errors='coerce')
    df_filtered.dropna(subset=['Year'], inplace=True) # Drop rows
where conversion failed
except Exception as e:
    print(f"Error converting 'Reference Date' to numeric: {e}. Please
check the format in your CSV.")
    print("Examples from data:", df_filtered['Reference
Date'].head().tolist())
    exit()


# Separate data by indicator
df_u5 = df_filtered[df_filtered['Indicator'] == 'Under-five mortality
rate'].copy()
df_neo = df_filtered[df_filtered['Indicator'] == 'Neonatal mortality
rate'].copy()

if df_u5.empty:
    print("No data found for 'Under-five mortality rate' after
filtering.")
if df_neo.empty:
    print("No data found for 'Neonatal mortality rate' after
filtering.")

# --- Prepare data for mapping (Latest Estimates per Country) ---
```

```
def get_latest_estimates(df_indicator):
    """Gets the latest observation for each country from the indicator
dataframe."""
    if df_indicator.empty:
        return pd.DataFrame()
    # Sort by year descending, then group by country and take the
first row (which is the latest)
    # Ensure the correct column name ('Observation Value') is used and
is numeric
    try:
        df_indicator['Observation Value'] =
pd.to_numeric(df_indicator['Observation Value'], errors='coerce')
        df_indicator.dropna(subset=['Observation Value'],
inplace=True) # Drop rows where OBS_VALUE is not numeric
    except Exception as e:
         print(f"Warning: Could not convert 'Observation Value' to
numeric for {df_indicator['Indicator'].iloc[0] if not
df_indicator.empty else 'some indicator'}: {e}")
         return pd.DataFrame() # Return empty if conversion fails

    # Sort by year descending, then by Observation Value descending
(for tie-breaking, though usually latest year is enough)
    latest_estimates = df_indicator.sort_values(by=['Year',
'Observation Value'], ascending=[False,
False]).groupby('REF_AREA').head(1).copy()
    # Select the correct column name for the value
    return latest_estimates[['REF_AREA', 'Geographic area',
'Observation Value', 'Year']]


# Use the corrected column name when calling get_latest_estimates and
storing results
latest_u5_map = get_latest_estimates(df_u5)
latest_neo_map = get_latest_estimates(df_neo)

if latest_u5_map.empty and latest_neo_map.empty:
    print("No latest 'Included in IGME' estimates could be determined
for mapping or identifying highest rates after filtering.")
    # Decide if you want to exit here or continue to plot empty graphs
    # exit()


# --- Visualize Latest Estimates on a Map ---

# Load shapefile
world = None
try:
    world = gpd.read_file(SHAPEFILE_PATH)
    print(f"Successfully loaded shapefile from {SHAPEFILE_PATH}")
```

```
    # Check for common ISO 3 code columns and select one
    shape_id_column = None
    if 'GID_0' in world.columns:
        shape_id_column = 'GID_0'
    elif 'ADM0_A3' in world.columns:
         shape_id_column = 'ADM0_A3'
    elif 'ISO' in world.columns: # <--- ADD THIS LINE
         shape_id_column = 'ISO'
    # Add more potential column names if needed based on your
shapefile source
    # elif 'your_shapefile_iso_col' in world.columns:
    #      shape_id_column = 'your_shapefile_iso_col'
    else:
         print("Error: Shapefile does not contain a recognized country
code column ('GID_0' or 'ADM0_A3').")
         print("Available columns in shapefile:",
world.columns.tolist())
         world = None # Disable mapping if column is missing


    if world is not None and shape_id_column is not None:
        # Filter world data to EAC countries for plotting efficiency
and focus
        world_eac =
world[world[shape_id_column].isin(EAC_COUNTRIES)].copy()
        if world_eac.empty:
            print("Warning: No matching EAC countries found in the
shapefile based on the provided codes.")
            print("Please check your SHAPEFILE_PATH and EAC_COUNTRIES
list. Country codes in shapefile:",
world[shape_id_column].unique().tolist()[:10]) # Print some samples
            world = None # Disable mapping if no matches were found
        else:
             # Rename the shapefile column to match the data column
for merging
            world_eac = world_eac.rename(columns={shape_id_column:
'REF_AREA'})
            # Ensure the 'Geographic area' column from the data is
available for merging
            # If it's not in world_eac already, we might need to merge
it before passing
            # However, the plot_map function merges the data, so the
name is in the data frame itself

except FileNotFoundError:
    print(f"Error: Shapefile not found at {SHAPEFILE_PATH}. Skipping
map visualization.")
    print("Please update SHAPEFILE_PATH with the correct location of
your downloaded shapefile and ensure all associated files (.shx, .dbf,
etc.) are present.")
```

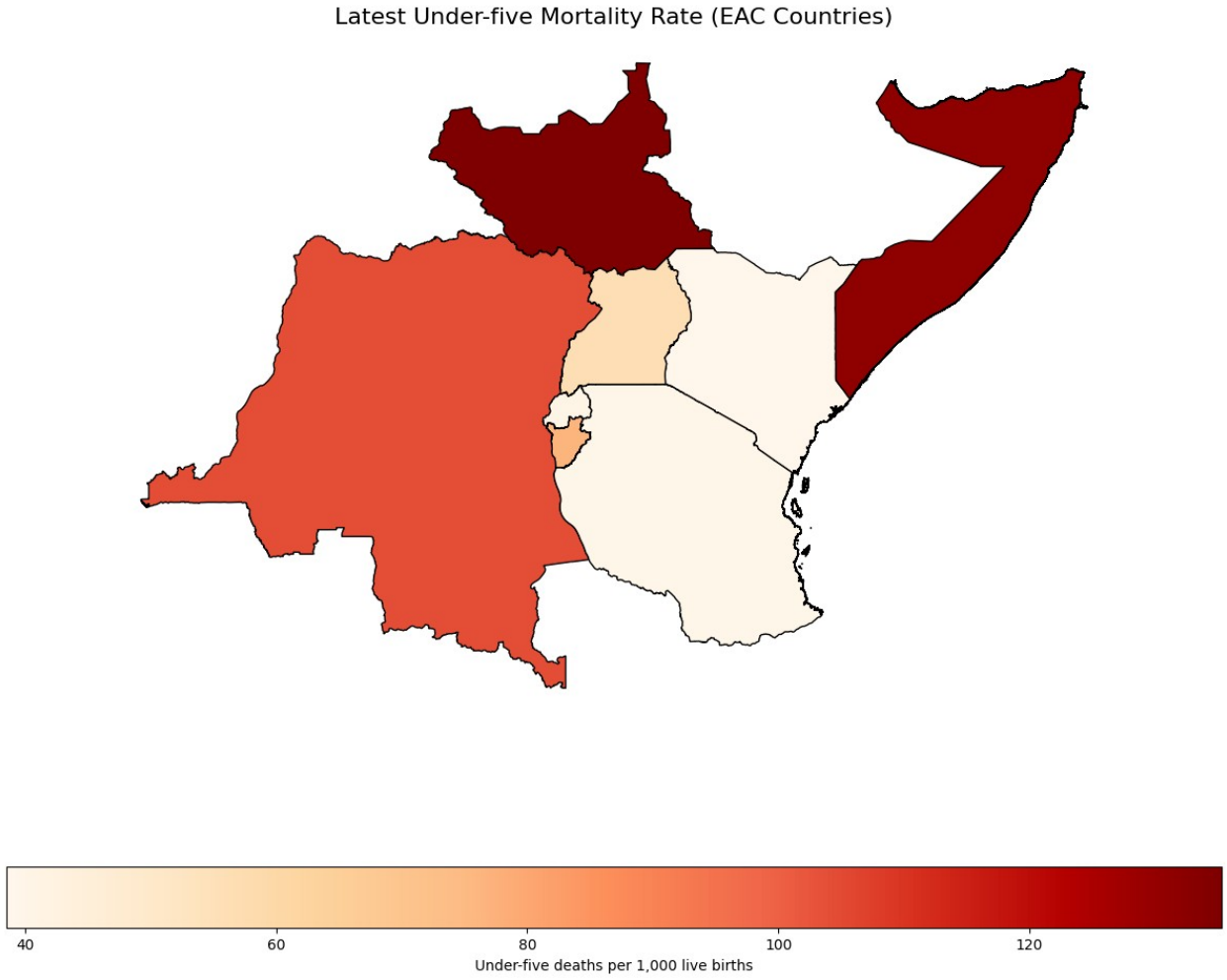
```
    world = None # Ensure world is None if file not found
except Exception as e:
    print(f"Error loading or processing shapefile: {e}. Skipping map
visualization.")
    print("Ensure you have installed geopandas and its dependencies
(like fiona/shapely) and that your shapefile is complete and not
corrupted.")
    world = None # Ensure world is None if error occurs


def plot_map(geodata, data_for_merge, title, value_column_name,
cmap='OrRd', legend_label='Rate'):
    """Plots geographic data colored by a data column."""
    # Check if geodata and data are valid before proceeding
    if geodata is None or data_for_merge.empty:
        print(f"Cannot plot map for '{title}' due to missing geodata
or data.")
        return

    # Merge data with geodata
    # Use the common 'REF_AREA' column after renaming in the loading
step
    merged = geodata.merge(data_for_merge, on='REF_AREA', how='left')

    # Handle cases where countries might be in EAC_COUNTRIES list but
not in the data_for_merge
    # (e.g., no 'Included in IGME' data for that indicator/country)
    countries_in_geo = set(geodata['REF_AREA'].unique())
    countries_with_data = set(data_for_merge['REF_AREA'].unique())
    missing_data_countries = list(countries_in_geo -
countries_with_data)

    if missing_data_countries:
         print(f"Note: No latest 'Included in IGME' data found for
mapping for countries in geodata: {',
'.join([EAC_COUNTRY_NAMES.get(code, code) for code in
missing_data_countries])}") # Use names for print

    fig, ax = plt.subplots(1, 1, figsize=(12, 10))

    # Plot the merged data, coloring by the specified column
    # Use the correct value_column_name ('Observation Value')
    if value_column_name in merged.columns:
        merged.plot(column=value_column_name, ax=ax, legend=True,
                    cmap=cmap,
                    missing_kwds={
                        'color': 'lightgrey',
                        'edgecolor': 'black',
                        'label': 'No latest Included in IGME data'
                    },
```

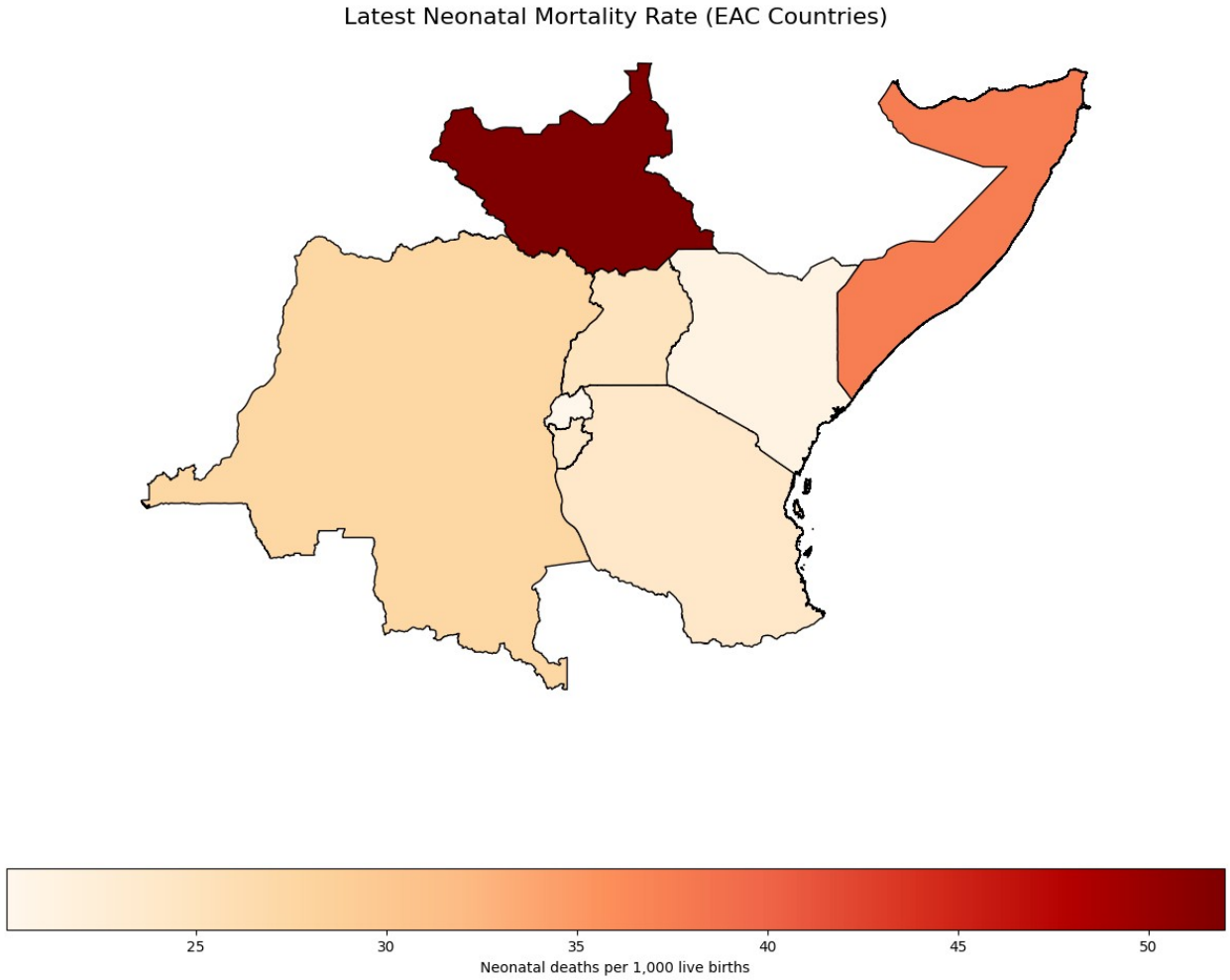
```
                    edgecolor='black',
                    legend_kwds={'label': legend_label, 'orientation':
"horizontal"})

        ax.set_title(title, fontsize=16)
        ax.set_axis_off()
        plt.tight_layout()
        plt.show()
    else:
        print(f"Error: Plotting column '{value_column_name}' not found
in merged data for map '{title}'.")


# Plot maps if shapefile was loaded successfully and filtered EAC
countries were found
if 'world_eac' in locals() and world_eac is not None and not
world_eac.empty:
    # Pass the correct value column name 'Observation Value' to
plot_map
    plot_map(world_eac, latest_u5_map, 'Latest Under-five Mortality
Rate (EAC Countries)', 'Observation Value', legend_label='Under-five
deaths per 1,000 live births')
    plot_map(world_eac, latest_neo_map, 'Latest Neonatal Mortality
Rate (EAC Countries)', 'Observation Value', legend_label='Neonatal
deaths per 1,000 live births')
else:
    print("\nMap visualization skipped due to shapefile
loading/filtering issues or no EAC countries found in shapefile.")


# --- Show Average Trends over Time and Country Points ---

def plot_trends(df_indicator, indicator_name):
    """Plots the average trend and individual country data points over
time."""
    if df_indicator.empty:
        print(f"No data to plot trends for '{indicator_name}'.")
        return

    # Ensure the correct column ('Observation Value') is numeric
before calculation and plotting
    try:
        df_indicator['Observation Value'] =
pd.to_numeric(df_indicator['Observation Value'], errors='coerce')
        df_indicator.dropna(subset=['Observation Value'],
inplace=True) # Drop rows where conversion failed
        if df_indicator.empty:
            print(f"No valid numeric 'Observation Value' data after
cleaning for '{indicator_name}'. Skipping plot.")
            return
```

```
    except Exception as e:
         print(f"Warning: Could not convert 'Observation Value' to
numeric for plotting trend {indicator_name}: {e}")
         print("Skipping trend plot.")
         return


    # Calculate average trend across selected countries for each year
    avg_trend = df_indicator.groupby('Year')['Observation
Value'].mean().reset_index()

    plt.figure(figsize=(14, 7))

    # Plot the average trend line
    plt.plot(avg_trend['Year'], avg_trend['Observation Value'],
label=f'Average Trend ({indicator_name} - EAC)', color='red',
linewidth=2)

    # Add individual country data points using the correct column
    plt.scatter(df_indicator['Year'], df_indicator['Observation
Value'], color='blue', alpha=0.5, label='Country Data Points (Included
in IGME)', s=15)

    plt.xlabel('Year (approx. midpoint)', fontsize=12)
    plt.ylabel('Deaths per 1,000 live births', fontsize=12)
    plt.title(f'{indicator_name} Trends (EAC Countries) with Average
Trend (Included in IGME data)', fontsize=16)
    plt.grid(True, linestyle='--', alpha=0.6)
    plt.legend(fontsize=10)

    # Improve x-axis ticks readability
    min_year = int(df_indicator['Year'].min()) if not
df_indicator['Year'].empty else 1990
    max_year = int(df_indicator['Year'].max()) if not
df_indicator['Year'].empty else 2025
    plt.xticks(range(min_year, max_year + 2, 5))
    plt.xlim(min_year - 1, max_year + 1)

    plt.show()

print("\n--- Plotting Trends ---")
plot_trends(df_u5, 'Under-five mortality rate')
plot_trends(df_neo, 'Neonatal mortality rate')


# --- Identify Countries with Highest Rates (Latest Year) ---

def identify_highest(latest_df, indicator_name):
    """Identifies the country with the highest latest rate for an
indicator."""
```

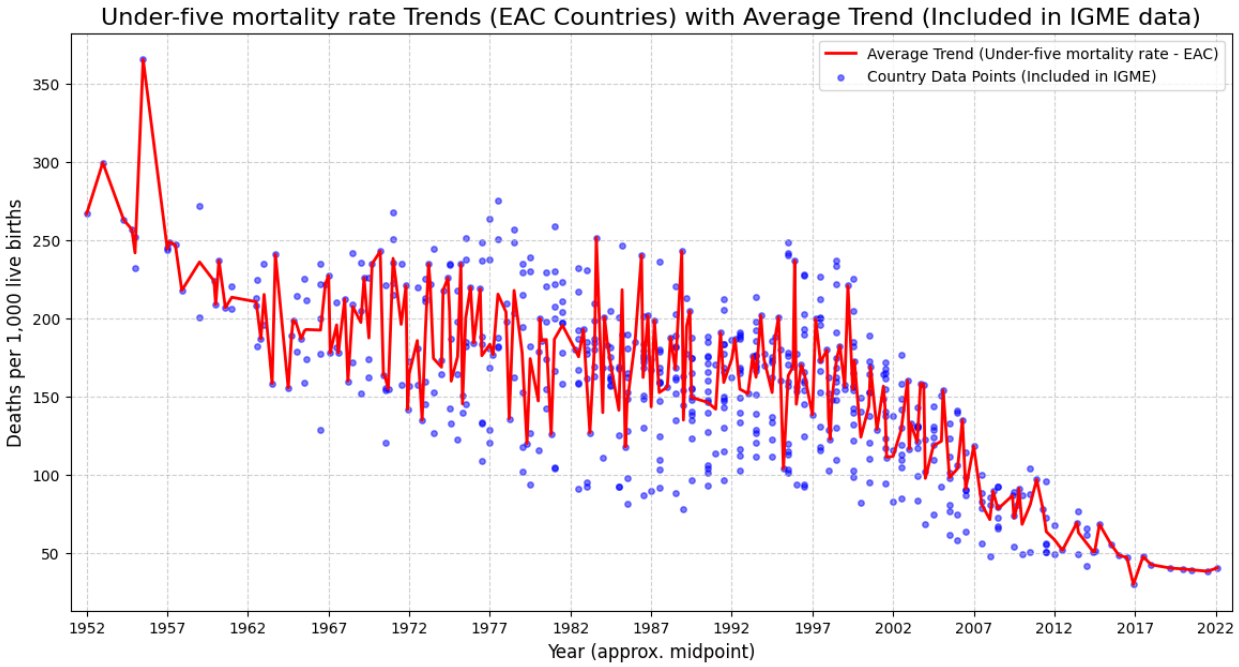
```
    if latest_df.empty:
        print(f"\nCould not identify highest country for
'{indicator_name}' due to missing data.")
        return

    # Ensure 'Observation Value' is numeric
    if not pd.api.types.is_numeric_dtype(latest_df['Observation
Value']):
         print(f"Error: 'Observation Value' is not numeric for
'{indicator_name}'. Cannot identify highest.")
         return

    # Find the country with the maximum 'Observation Value'
    highest_country_row_index = latest_df['Observation
Value'].idxmax()
    highest_country = latest_df.loc[highest_country_row_index]

    print(f"\nCountry with the highest latest '{indicator_name}'
({highest_country['Year']:.1f}):")
    # Use the correct column name 'Observation Value' for the rate
    print(f"- {EAC_COUNTRY_NAMES.get(highest_country['REF_AREA'],
highest_country['Geographic area'])} ({highest_country['REF_AREA']}):
{highest_country['Observation Value']:.2f} deaths per 1,000 live
births.")


print("\n--- Identifying Highest Latest Rates ---")
# Pass the latest dataframes, which should now contain the
'Observation Value' column
identify_highest(latest_u5_map, 'Under-five mortality rate')
identify_highest(latest_neo_map, 'Neonatal mortality rate')

print("\nAnalysis complete.")
```

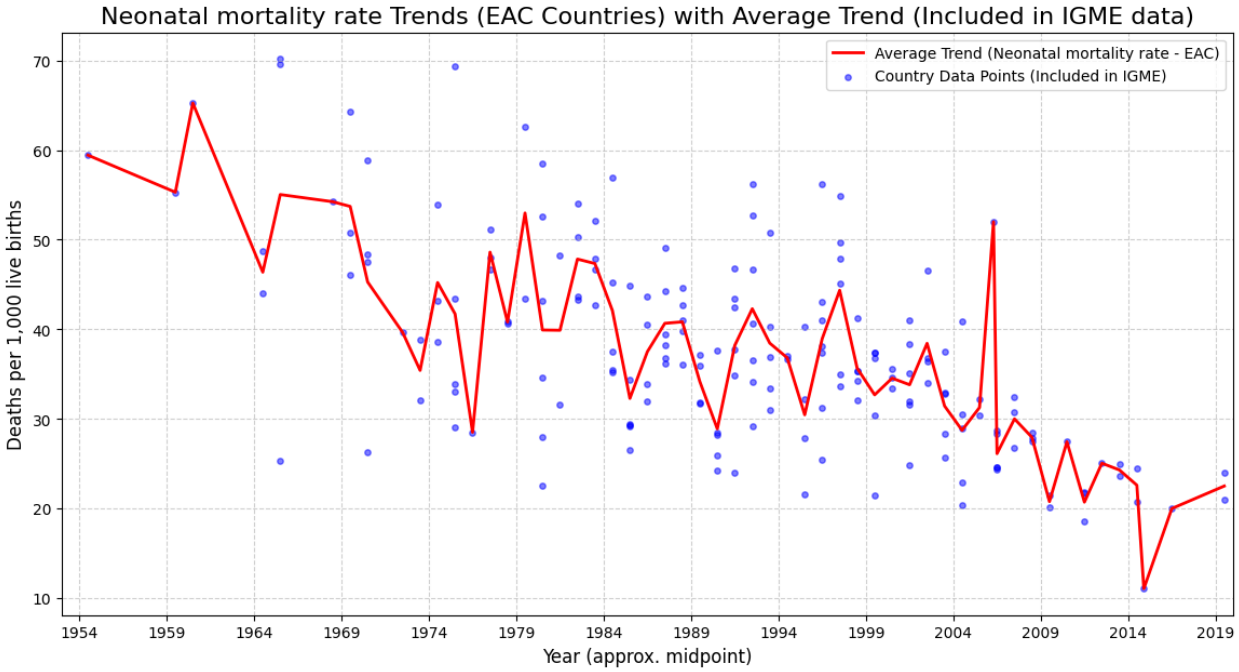
```
Successfully loaded neonatal_datascience.csv
Filtered data includes 805 rows for EAC countries, Total Sex/Wealth,
Included in IGME.
Successfully loaded shapefile from Africa_Boundaries.shp
```


Latest Under-five Mortality Rate (EAC Countries)
40
60
80
100
120
Under-five deaths per 1,000 live births




```
--- Plotting Trends ---
```






```
--- Identifying Highest Latest Rates ---

Country with the highest latest 'Under-five mortality rate' (2006.3):
- South Sudan (SSD): 135.30 deaths per 1,000 live births.

Country with the highest latest 'Neonatal mortality rate' (2006.3):
- South Sudan (SSD): 52.00 deaths per 1,000 live births.
```

```
Analysis complete.
```