healthcare-dash-colab.ipynb ☆

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

RAM Disk

✦ Gemini

[14]
```
# Ensure necessary libraries are installed
!pip install -U  kaleido dash
import plotly.express as px
import pandas as pd
import requests
from dash import Dash, dcc, html
from dash.dependencies import Input, Output
```

```
Requirement already satisfied: kaleido in /usr/local/lib/python3.10/dist-packages (0.2.1)
Requirement already satisfied: dash in /usr/local/lib/python3.10/dist-packages (2.17.1)
Requirement already satisfied: Flask<3.1,>=1.0.4 in /usr/local/lib/python3.10/dist-packages (from dash) (2.2.5)
Requirement already satisfied: Werkzeug<3.1 in /usr/local/lib/python3.10/dist-packages (from dash) (3.0.3)
Requirement already satisfied: plotly>=5.0.0 in /usr/local/lib/python3.10/dist-packages (from dash) (5.22.0)
Requirement already satisfied: dash-html-components==2.0.0 in /usr/local/lib/python3.10/dist-packages (from dash) (2.0.0)
Requirement already satisfied: dash-core-components==2.0.0 in /usr/local/lib/python3.10/dist-packages (from dash) (2.0.0)
Requirement already satisfied: dash-table==5.0.0 in /usr/local/lib/python3.10/dist-packages (from dash) (5.0.0)
Requirement already satisfied: importlib-metadata in /usr/local/lib/python3.10/dist-packages (from dash) (8.0.0)
Requirement already satisfied: typing-extensions>=4.1.1 in /usr/local/lib/python3.10/dist-packages (from dash) (4.12.2)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (from dash) (2.31.0)
Requirement already satisfied: retrying in /usr/local/lib/python3.10/dist-packages (from dash) (1.3.4)
Requirement already satisfied: nest-asyncio in /usr/local/lib/python3.10/dist-packages (from dash) (1.6.0)
Requirement already satisfied: setuptools in /usr/local/lib/python3.10/dist-packages (from dash) (67.7.2)
Requirement already satisfied: Jinja2>=3.0 in /usr/local/lib/python3.10/dist-packages (from Flask<3.1,>=1.0.4->dash) (3.1.4)
Requirement already satisfied: itsdangerous>=2.0 in /usr/local/lib/python3.10/dist-packages (from Flask<3.1,>=1.0.4->dash) (2.2.0)
Requirement already satisfied: click>=8.0 in /usr/local/lib/python3.10/dist-packages (from Flask<3.1,>=1.0.4->dash) (8.1.7)
Requirement already satisfied: tenacity>=6.2.0 in /usr/local/lib/python3.10/dist-packages (from plotly>=5.0.0->dash) (8.5.0)
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-packages (from plotly>=5.0.0->dash) (24.1)
Requirement already satisfied: MarkupSafe>=2.1.1 in /usr/local/lib/python3.10/dist-packages (from Werkzeug<3.1->dash) (2.1.5)
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.10/dist-packages (from importlib-metadata->dash) (3.19.2)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests->dash) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests->dash) (3.7)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests->dash) (2.0.7)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests->dash) (2024.7.4)
Requirement already satisfied: six>=1.7.0 in /usr/local/lib/python3.10/dist-packages (from retrying->dash) (1.16.0)
```

```
# Define CDC API endpoints
CDC_BASE_URL = "https://data.cdc.gov/resource/"
CDC_API_ENDPOINTS = {
    "mortality": "xbxb-epbu.json",  # Example endpoint for mortality data
    "chronic_diseases": "5h56-n989.json",  # Example endpoint for chronic diseases
}

# Define WHO API endpoints
WHO_BASE_URL = "https://ghoapi.azureedge.net/api/"
WHO_API_ENDPOINTS = {
    "life_expectancy": "WHOSIS_000001",  # Example endpoint for life expectancy
}
```

[16]
```
# Function to fetch data from CDC API
def fetch_cdc_data(endpoint, params=None):
    url = f"{CDC_BASE_URL}{endpoint}"
    response = requests.get(url, params=params)
    response.raise_for_status()  # Raise an exception for HTTP errors
    return pd.DataFrame(response.json())

# Function to fetch data from WHO API
def fetch_who_data(endpoint, params=None):
    url = f"{WHO_BASE_URL}{endpoint}"
    response = requests.get(url, params=params)
    response.raise_for_status()  # Raise an exception for HTTP errors
    return pd.DataFrame(response.json()['value'])

# Fetching metadata from CDC mortality data endpoint
cdc_mortality_metadata = fetch_cdc_data(CDC_API_ENDPOINTS["mortality"], {"$limit": 1})
print("CDC Mortality Metadata:")
print(cdc_mortality_metadata.columns)
print(cdc_mortality_metadata.dtypes)

# Fetching metadata from WHO life expectancy data endpoint
who_life_expectancy_metadata = fetch_who_data(WHO_API_ENDPOINTS["life_expectancy"], {"$top": 1})
print("WHO Life Expectancy Metadata:")
print(who_life_expectancy_metadata.columns)
print(who_life_expectancy_metadata.dtypes)
```

```
CDC Mortality Metadata:
Index(['state', 'year', 'sex', 'age_group', 'race_and_hispanic_origin',
       'deaths', 'population', 'crude_death_rate',
       'standard_error_for_crude_rate',
       'lower_confidence_limit_for_crude_rate',
       'upper_confidence_limit_for_crude_rate', 'age_adjusted_rate',
       'standard_error_for_age_adjusted_rate',
       'lower_confidence_limit_for_age_adjusted_rate',
       'upper_confidence_limit_for_age_adjusted_rate',
       'state_crude_rate_in_range', 'us_crude_rate', 'us_age_adjusted_rate',
       'unit'],
      dtype='object')
state                                   object
year                                    object
sex                                     object
age_group                               object
race_and_hispanic_origin                object
deaths                                  object
population                              object
crude_death_rate                        object
standard_error_for_crude_rate           object
lower_confidence_limit_for_crude_rate   object
upper_confidence_limit_for_crude_rate   object
```

```
age_adjusted_rate                               object
standard_error_for_age_adjusted_rate            object
lower_confidence_limit_for_age_adjusted_rate    object
upper_confidence_limit_for_age_adjusted_rate    object
state_crude_rate_in_range                       object
us_crude_rate                                   object
us_age_adjusted_rate                            object
unit                                            object
dtype: object
WHO Life Expectancy Metadata:
Index(['Id', 'IndicatorCode', 'SpatialDimType', 'SpatialDim', 'TimeDimType',
       'ParentLocationCode', 'ParentLocation', 'Dim1Type', 'Dim1', 'TimeDim',
       'Dim2Type', 'Dim2', 'Dim3Type', 'Dim3', 'DataSourceDimType',
       'DataSourceDim', 'Value', 'NumericValue', 'Low', 'High', 'Comments',
       'Date', 'TimeDimensionValue', 'TimeDimensionBegin', 'TimeDimensionEnd'],
      dtype='object')
Id                   int64
IndicatorCode       object
SpatialDimType      object
SpatialDim          object
TimeDimType         object
ParentLocationCode  object
ParentLocation      object
Dim1Type            object
Dim1                object
TimeDim              int64
Dim2Type            object
Dim2                object
Dim3Type            object
Dim3                object
DataSourceDimType   object
DataSourceDim       object
Value               object
NumericValue       float64
Low                 object
```

[17]
```python
# Example data cleaning steps for CDC mortality data
def clean_cdc_data(df):
    # Convert 'year' to datetime
    df['year'] = pd.to_datetime(df['year'], format='%Y')
    # Filter out unwanted columns
    df = df[['year', 'state', 'sex', 'age_group', 'race_and_hispanic_origin', 'deaths', 'age_adjusted_rate']]
    # Handle missing values
    df.dropna(inplace=True)
    return df

# Example data cleaning steps for WHO life expectancy data
def clean_who_data(df):
    # Convert 'TimeDimensionBegin' and 'TimeDimensionEnd' to datetime
    df['TimeDimensionBegin'] = pd.to_datetime(df['TimeDimensionBegin'])
    df['TimeDimensionEnd'] = pd.to_datetime(df['TimeDimensionEnd'])
    # Filter out unwanted columns
    df = df[['TimeDimensionBegin', 'TimeDimensionEnd', 'SpatialDim', 'Dim1', 'NumericValue']]
    df = df.rename(columns={'TimeDimensionBegin': 'start_date', 'TimeDimensionEnd': 'end_date', 'SpatialDim': 'country', 'Dim1': 'sex', 'NumericValue': 'life_expectancy'})
    # Handle missing values
    df.dropna(inplace=True)
    return df
```

[18]
```python
cdc_mortality_data = fetch_cdc_data(CDC_API_ENDPOINTS["mortality"])
cdc_mortality_data_cleaned = clean_cdc_data(cdc_mortality_data)
print("Cleaned CDC Mortality Data:")
print(cdc_mortality_data_cleaned.head())

who_life_expectancy_data = fetch_who_data(WHO_API_ENDPOINTS["life_expectancy"])
who_life_expectancy_data_cleaned = clean_who_data(who_life_expectancy_data)
print("Cleaned WHO Life Expectancy Data:")
print(who_life_expectancy_data_cleaned.head())
```

```
Cleaned CDC Mortality Data:
        year    state         sex age_group race_and_hispanic_origin deaths  \
0 1999-01-01  Alabama  Both Sexes  All Ages    All Races-All Origins    169
1 2000-01-01  Alabama  Both Sexes  All Ages    All Races-All Origins    197
2 2001-01-01  Alabama  Both Sexes  All Ages    All Races-All Origins    216
3 2002-01-01  Alabama  Both Sexes  All Ages    All Races-All Origins    211
4 2003-01-01  Alabama  Both Sexes  All Ages    All Races-All Origins    197

  age_adjusted_rate
0            3.8521
1            4.4857
2            4.8915
3            4.7619
4            4.4333
Cleaned WHO Life Expectancy Data:
                start_date                  end_date country       sex  \
0 2015-01-01 00:00:00+01:00 2015-12-31 00:00:00+01:00     URY  SEX_BTSX
1 2019-01-01 00:00:00+01:00 2019-12-31 00:00:00+01:00     TON  SEX_BTSX
2 2015-01-01 00:00:00+01:00 2015-12-31 00:00:00+01:00  GLOBAL  SEX_BTSX
3 2000-01-01 00:00:00+01:00 2000-12-31 00:00:00+01:00     SUR  SEX_BTSX
4 2000-01-01 00:00:00+01:00 2000-12-31 00:00:00+01:00     GRC   SEX_MLE

   life_expectancy
0         77.03117
1         72.57066
2         72.27778
3         69.94162
4         75.51034
<ipython-input-17-dd8840733634>:8: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```

[19]
```python
# Function to save figures as JPG files using kaleido
def save_fig(fig, filename):
    fig.write_image(f"{filename}.jpg", engine='kaleido')
```

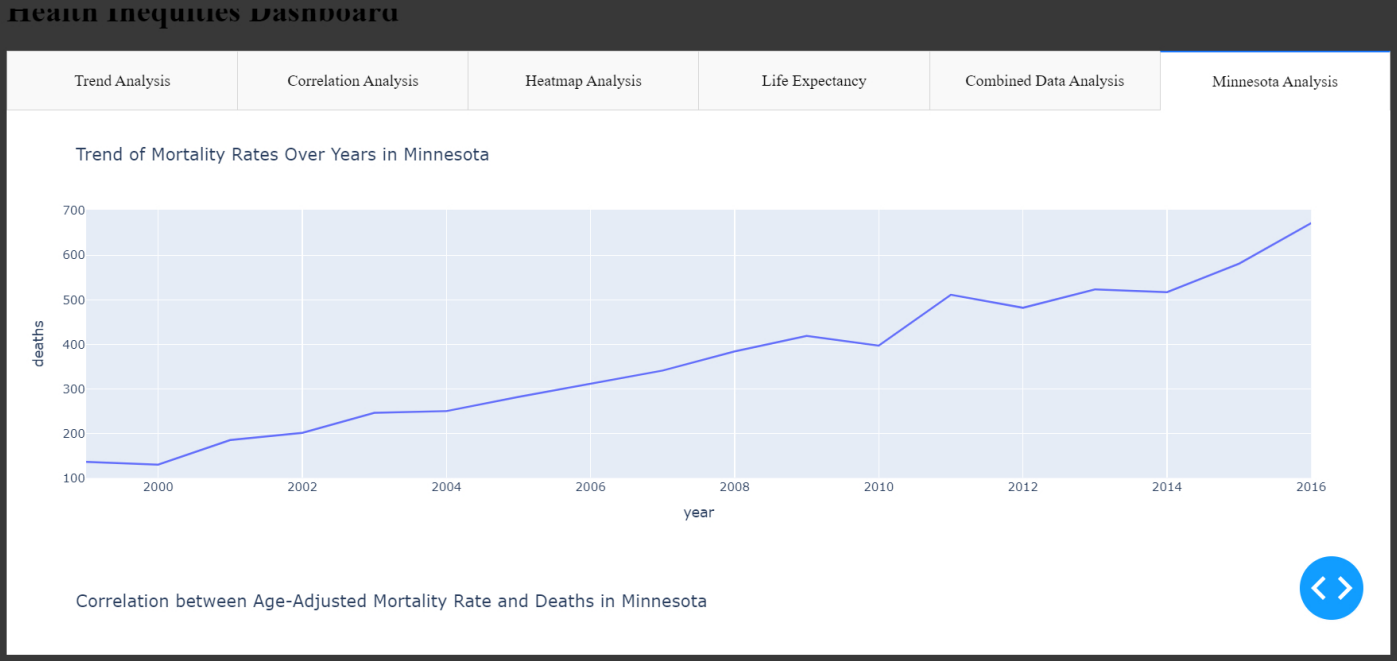
```python
# Generate and save CDC mortality data trend analysis
cdc_mortality_data_cleaned['year'] = pd.to_datetime(cdc_mortality_data_cleaned['year'], format='%Y')
cdc_mortality_data_cleaned['year'] = cdc_mortality_data_cleaned['year'].dt.year
cdc_mortality_data_cleaned['deaths'] = pd.to_numeric(cdc_mortality_data_cleaned['deaths'], errors='coerce')
mortality_trend = cdc_mortality_data_cleaned.groupby('year')['deaths'].sum().reset_index()
fig_mortality_trend = px.line(mortality_trend, x='year', y='deaths', title='Trend of Mortality Rates Over Years')
save_fig(fig_mortality_trend, 'mortality_trend')

# Generate and save correlation analysis between age-adjusted mortality rate and deaths
cdc_mortality_data_cleaned['age_adjusted_rate'] = pd.to_numeric(cdc_mortality_data_cleaned['age_adjusted_rate'], errors='coerce')
cdc_mortality_data_cleaned = cdc_mortality_data_cleaned.dropna(subset=['age_adjusted_rate', 'deaths'])
correlation = cdc_mortality_data_cleaned[['age_adjusted_rate', 'deaths']].corr()
fig_correlation = px.scatter(cdc_mortality_data_cleaned, x='age_adjusted_rate', y='deaths', trendline='ols', title='Correlation between Age-Adjusted Mortality Rate and Deaths')
save_fig(fig_correlation, 'correlation_analysis')

# Generate and save heatmap for age-adjusted mortality rates
fig_heatmap = px.density_heatmap(cdc_mortality_data_cleaned, x='year', y='state', z='age_adjusted_rate', title='Heatmap of Age-Adjusted Mortality Rates by State and Year')
save_fig(fig_heatmap, 'age_adjusted_mortality_heatmap')

# Generate and save average life expectancy by country
avg_life_expectancy = who_life_expectancy_data_cleaned.groupby('country')['life_expectancy'].mean().reset_index()
fig_life_expectancy = px.bar(avg_life_expectancy, x='country', y='life_expectancy', title='Average Life Expectancy by Country')
save_fig(fig_life_expectancy, 'average_life_expectancy')

# Generate and save combined data analysis of deaths and life expectancy
who_life_expectancy_data_cleaned['year'] = pd.to_datetime(who_life_expectancy_data_cleaned['start_date']).dt.year
combined_data = pd.merge(cdc_mortality_data_cleaned, who_life_expectancy_data_cleaned, on='year', how='inner')
combined_data = combined_data[['year', 'state', 'country', 'deaths', 'life_expectancy']]
fig_combined = px.scatter(combined_data, x='deaths', y='life_expectancy', color='state', title='Combined Analysis of Deaths and Life Expectancy')
save_fig(fig_combined, 'combined_data_analysis')
```

```
<ipython-input-20-6f676089f026>:2: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

<ipython-input-20-6f676089f026>:3: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

<ipython-input-20-6f676089f026>:4: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

<ipython-input-20-6f676089f026>:10: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```

```python
# Import necessary libraries
from dash import Dash, dcc, html
import plotly.express as px
import pandas as pd
from dash.dependencies import Input, Output

# Initialize the Dash app
app = Dash(__name__)

# Filter data for Minnesota
cdc_minnesota_data = cdc_mortality_data_cleaned[cdc_mortality_data_cleaned['state'] == 'Minnesota']
who_minnesota_data = who_life_expectancy_data_cleaned[who_life_expectancy_data_cleaned['country'] == 'United States']

# Ensure 'year' column is in datetime format and extract year for Minnesota data
cdc_minnesota_data['year'] = pd.to_datetime(cdc_minnesota_data['year'], format='%Y').dt.year

# Convert 'deaths' column to numeric if not already
cdc_minnesota_data['deaths'] = pd.to_numeric(cdc_minnesota_data['deaths'], errors='coerce')

# Group by year and calculate the sum of deaths for each year for Minnesota
mortality_trend_minnesota = cdc_minnesota_data.groupby('year')['deaths'].sum().reset_index()

# Create figure for trend analysis for Minnesota
fig_mortality_trend_minnesota = px.line(mortality_trend_minnesota, x='year', y='deaths', title='Trend of Mortality Rates Over Years in Minnesota')

# Analyze correlation between deaths and age_adjusted_rate for Minnesota
correlation_minnesota = cdc_minnesota_data[['age_adjusted_rate', 'deaths']].corr()

# Create scatter plot for visualizing correlation for Minnesota
fig_correlation_minnesota = px.scatter(cdc_minnesota_data, x='age_adjusted_rate', y='deaths', trendline='ols',
                                       title='Correlation between Age-Adjusted Mortality Rate and Deaths in Minnesota')

# Plot heatmap for age-adjusted mortality rates for Minnesota
fig_heatmap_minnesota = px.density_heatmap(cdc_minnesota_data, x='year', y='age_group', z='age_adjusted_rate',
                                           title='Heatmap of Age-Adjusted Mortality Rates in Minnesota by Age Group and Year')

# Calculate average life expectancy by year for Minnesota
avg_life_expectancy_minnesota = who_minnesota_data.groupby('year')['life_expectancy'].mean().reset_index()

# Plot bar chart for average life expectancy by year for Minnesota
fig_life_expectancy_minnesota = px.bar(avg_life_expectancy_minnesota, x='year', y='life_expectancy', title='Average Life Expectancy by Year in Minnesota')

# Display combined analysis of deaths and life expectancy for Minnesota
```

```python
combined_data_minnesota = pd.merge(cdc_minnesota_data, who_minnesota_data, left_on='year', right_on='year', how='inner')
combined_data_minnesota = combined_data_minnesota[['year', 'deaths', 'life_expectancy']]

fig_combined_minnesota = px.scatter(combined_data_minnesota, x='deaths', y='life_expectancy', title='Combined Analysis of Deaths and Life Expectancy in Minnesota')

# Define the layout of the app
app.layout = html.Div([
    html.H1("Health Inequities Dashboard"),
    dcc.Tabs([
        dcc.Tab(label='Trend Analysis', children=[
            dcc.Graph(id='mortality-trend-graph', figure=fig_mortality_trend)
        ]),
        dcc.Tab(label='Correlation Analysis', children=[
            dcc.Graph(id='correlation-graph', figure=fig_correlation)
        ]),
        dcc.Tab(label='Heatmap Analysis', children=[
            dcc.Dropdown(
                id='state-dropdown',
                options=[{'label': state, 'value': state} for state in cdc_mortality_data_cleaned['state'].unique()],
                value=cdc_mortality_data_cleaned['state'].unique()[0]
            ),
            dcc.Graph(id='heatmap-graph')
        ]),
        dcc.Tab(label='Life Expectancy', children=[
            dcc.Graph(id='life-expectancy-graph', figure=fig_life_expectancy)
        ]),
        dcc.Tab(label='Combined Data Analysis', children=[
            dcc.Graph(id='combined-data-graph', figure=fig_combined)
        ]),
        dcc.Tab(label='Minnesota Analysis', children=[
            dcc.Graph(id='mortality-trend-graph-minnesota', figure=fig_mortality_trend_minnesota),
            dcc.Graph(id='correlation-graph-minnesota', figure=fig_correlation_minnesota),
            dcc.Graph(id='heatmap-graph-minnesota', figure=fig_heatmap_minnesota),
            dcc.Graph(id='life-expectancy-graph-minnesota', figure=fig_life_expectancy_minnesota),
            dcc.Graph(id='combined-data-graph-minnesota', figure=fig_combined_minnesota)
        ])
    ])
])

@app.callback(
    Output('heatmap-graph', 'figure'),
    Input('state-dropdown', 'value')
)
def update_heatmap(selected_state):
    filtered_data = cdc_mortality_data_cleaned[cdc_mortality_data_cleaned['state'] == selected_state]
    fig_heatmap = px.density_heatmap(filtered_data, x='year', y='age_group', z='age_adjusted_rate',
                                     title=f'Heatmap of Age-Adjusted Mortality Rates in {selected_state}')
    return fig_heatmap

# Run the app
if __name__ == '__main__':
    app.run_server(debug=True)
```

```
<ipython-input-21-b7668ae42592>:15: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

<ipython-input-21-b7668ae42592>:18: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```




```python
# Generate and save Minnesota-specific visualizations
```

```python
# Filter data for Minnesota
cdc_minnesota_data = cdc_mortality_data_cleaned[cdc_mortality_data_cleaned['state'] == 'Minnesota']
who_minnesota_data = who_life_expectancy_data_cleaned[who_life_expectancy_data_cleaned['country'] == 'United States']

# Ensure 'year' column is in datetime format and extract year for Minnesota data
cdc_minnesota_data['year'] = pd.to_datetime(cdc_minnesota_data['year'], format='%Y').dt.year

# Convert 'deaths' column to numeric if not already
cdc_minnesota_data['deaths'] = pd.to_numeric(cdc_minnesota_data['deaths'], errors='coerce')

# Group by year and calculate the sum of deaths for each year for Minnesota
mortality_trend_minnesota = cdc_minnesota_data.groupby('year')['deaths'].sum().reset_index()

# Create figure for trend analysis for Minnesota
fig_mortality_trend_minnesota = px.line(mortality_trend_minnesota, x='year', y='deaths', title='Trend of Mortality Rates Over Years in Minnesota')
fig_mortality_trend_minnesota.write_image("mortality_trend_minnesota.jpg")

# Analyze correlation between deaths and age_adjusted_rate for Minnesota
cdc_minnesota_data['age_adjusted_rate'] = pd.to_numeric(cdc_minnesota_data['age_adjusted_rate'], errors='coerce')
cdc_minnesota_data = cdc_minnesota_data.dropna(subset=['age_adjusted_rate', 'deaths'])
correlation_minnesota = cdc_minnesota_data[['age_adjusted_rate', 'deaths']].corr()

# Create scatter plot for visualizing correlation for Minnesota
fig_correlation_minnesota = px.scatter(cdc_minnesota_data, x='age_adjusted_rate', y='deaths', trendline='ols', title='Correlation between Age-Adjusted Mortality Rate and Deaths in Minn
fig_correlation_minnesota.write_image("correlation_analysis_minnesota.jpg")

# Plot heatmap for age-adjusted mortality rates for Minnesota
fig_heatmap_minnesota = px.density_heatmap(cdc_minnesota_data, x='year', y='age_group', z='age_adjusted_rate', title='Heatmap of Age-Adjusted Mortality Rates in Minnesota by Age Group
fig_heatmap_minnesota.write_image("heatmap_minnesota.jpg")

# Calculate average life expectancy by year for Minnesota
avg_life_expectancy_minnesota = who_minnesota_data.groupby('year')['life_expectancy'].mean().reset_index()

# Plot bar chart for average life expectancy by year for Minnesota
fig_life_expectancy_minnesota = px.bar(avg_life_expectancy_minnesota, x='year', y='life_expectancy', title='Average Life Expectancy by Year in Minnesota')
fig_life_expectancy_minnesota.write_image("life_expectancy_minnesota.jpg")

# Display combined analysis of deaths and life expectancy for Minnesota
combined_data_minnesota = pd.merge(cdc_minnesota_data, who_minnesota_data, left_on='year', right_on='year', how='inner')
combined_data_minnesota = combined_data_minnesota[['year', 'deaths', 'life_expectancy']]

fig_combined_minnesota = px.scatter(combined_data_minnesota, x='deaths', y='life_expectancy', title='Combined Analysis of Deaths and Life Expectancy in Minnesota')
fig_combined_minnesota.write_image("combined_data_analysis_minnesota.jpg")
```

```
<ipython-input-24-4dfa3ccd181c>:7: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

<ipython-input-24-4dfa3ccd181c>:10: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

<ipython-input-24-4dfa3ccd181c>:20: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```

```
[25] !jupyter nbconvert --to html your_notebook.ipynb
```

```
[NbConvertApp] WARNING | pattern 'your_notebook.ipynb' matched no files
This application is used to convert notebook files (*.ipynb)
        to various other formats.

        WARNING: THE COMMANDLINE INTERFACE MAY CHANGE IN FUTURE RELEASES.

Options
=======
The options below are convenience aliases to configurable class-options,
as listed in the "Equivalent to" description-line of the aliases.
To see all configurable class-options for some <cmd>, use:
    <cmd> --help-all

--debug
    set log level to logging.DEBUG (maximize logging output)
    Equivalent to: [--Application.log_level=10]
--show-config
    Show the application's configuration (human-readable format)
    Equivalent to: [--Application.show_config=True]
--show-config-json
    Show the application's configuration (json format)
    Equivalent to: [--Application.show_config_json=True]
--generate-config
    generate default config file
    Equivalent to: [--JupyterApp.generate_config=True]
-y
    Answer yes to any questions instead of prompting.
    Equivalent to: [--JupyterApp.answer_yes=True]
--execute
    Execute the notebook prior to export.
    Equivalent to: [--ExecutePreprocessor.enabled=True]
--allow-errors
    Continue notebook execution even if one of the cells throws an error and include the error message in the cell output (the default behaviour is to abort conversion). This flag is
    Equivalent to: [--ExecutePreprocessor.allow_errors=True]
--stdin
    read a single notebook file from stdin. Write the resulting notebook with default basename 'notebook.*'
    Equivalent to: [--NbConvertApp.from_stdin=True]
--stdout
    Write notebook output to stdout instead of files.
```

```
    Write notebook output to stdout instead of files.
    Equivalent to: [--NbConvertApp.writer_class=StdoutWriter]
--inplace
    Run nbconvert in place, overwriting the existing notebook (only
            relevant when converting to notebook format)
    Equivalent to: [--NbConvertApp.use_output_suffix=False --NbConvertApp.export_format=notebook --FilesWriter.build_directory=]
--clear-output
    Clear output of current file and save in place,
            overwriting the existing notebook.
    Equivalent to: [--NbConvertApp.use_output_suffix=False --NbConvertApp.export_format=notebook --FilesWriter.build_directory= --ClearOutputPreprocessor.enabled=True]
--no-prompt
    Exclude input and output prompts from converted document.
    Equivalent to: [--TemplateExporter.exclude_input_prompt=True --TemplateExporter.exclude_output_prompt=True]
--no-input
    Exclude input cells and output prompts from converted document.
            This mode is ideal for generating code-free reports.
    Equivalent to: [--TemplateExporter.exclude_output_prompt=True --TemplateExporter.exclude_input=True --TemplateExporter.exclude_input_prompt=True]
--allow-chromium-download
    Whether to allow downloading chromium if no suitable version is found on the system.
```

[ ] Start coding or generate with AI.

✓ 3s completed at 02:25