Beda.Milestone.2

December 22, 2024

[8]:
```
import pandas as pd

# Define the correct file path
file_path = '/Users/cheribeda/Desktop/nationaldatabaseofchildcareprices (1).
 ↪xlsx'

# Load the Excel file into a Pandas DataFrame
try:
    data = pd.read_excel(file_path)
    # Display the first few rows of the dataset
    print("File loaded successfully!")
    print(data.head())
except FileNotFoundError:
    print("File not found. Please check the file path.")
except Exception as e:
    print(f"An error occurred: {e}")
```

```
File loaded successfully!
  State_Name State_Abbreviation     County_Name  County_FIPS_Code  StudyYear  \
0    Alabama                 AL  Autauga County              1001       2008
1    Alabama                 AL  Autauga County              1001       2009
2    Alabama                 AL  Autauga County              1001       2010
3    Alabama                 AL  Autauga County              1001       2011
4    Alabama                 AL  Autauga County              1001       2012

   UNR_16  FUNR_16  MUNR_16  UNR_20to64  FUNR_20to64  …  MFCCToddler  \
0    5.42     4.41     6.32         4.6          3.5  …        83.45
1    5.93     5.72     6.11         4.8          4.6  …        87.39
2    6.21     5.57     6.78         5.1          4.6  …        91.33
3    7.55     8.13     7.03         6.2          6.3  …        95.28
4    8.60     8.88     8.29         6.7          6.4  …        99.22

   MFCCToddler_flag  MFCCPreschool  MFCCPreschool_flag  _75FCCInfant  \
0               3.0          81.40                 1.0          97.4
1               3.0          85.68                 1.0         102.0
2               3.0          89.96                 1.0         106.6
3               3.0          94.25                 1.0         111.2
4               3.0          98.53                 1.0         115.8
```

```
   _75FCCInfant_flag  _75FCCToddler  _75FCCToddler_flag  _75FCCPreschool  \
0                1.0           97.4                 3.0             95.0
1                1.0          102.0                 3.0            100.0
2                1.0          106.6                 3.0            105.0
3                1.0          111.2                 3.0            110.0
4                1.0          115.8                 3.0            115.0

   _75FCCPreschool_flag
0                   1.0
1                   1.0
2                   1.0
3                   1.0
4                   1.0

[5 rows x 227 columns]
```

[10]:
```
# Get the shape of the dataset
print("Dataset shape:", data.shape)

# Display all column names
print("Column names:")
print(data.columns)

# Check for missing values
print("Missing values per column:")
print(data.isnull().sum())
```

```
Dataset shape: (34567, 227)
Column names:
Index(['State_Name', 'State_Abbreviation', 'County_Name', 'County_FIPS_Code',
       'StudyYear', 'UNR_16', 'FUNR_16', 'MUNR_16', 'UNR_20to64',
       'FUNR_20to64',
       …
       'MFCCToddler', 'MFCCToddler_flag', 'MFCCPreschool',
       'MFCCPreschool_flag', '_75FCCInfant', '_75FCCInfant_flag',
       '_75FCCToddler', '_75FCCToddler_flag', '_75FCCPreschool',
       '_75FCCPreschool_flag'],
      dtype='object', length=227)
Missing values per column:
State_Name                  0
State_Abbreviation          0
County_Name                 0
County_FIPS_Code            0
StudyYear                   0
                           …
_75FCCInfant_flag       11184
_75FCCToddler           11184
```

```
_75FCCToddler_flag      11184
_75FCCPreschool         11184
_75FCCPreschool_flag    11184
Length: 227, dtype: int64
```

[12]:
```
# Get a statistical summary of numeric columns
print(data.describe())

# Check the data types of all columns
print("Data types:")
print(data.dtypes)
```

```
       County_FIPS_Code     StudyYear        UNR_16      FUNR_16  \
count      34567.000000  34567.000000  34567.000000  34567.00000
mean       30388.132786   2012.999711      7.465902      7.02902
std        15161.015383      3.162232      3.538619      3.56342
min         1001.000000   2008.000000      0.000000      0.00000
25%        18177.000000   2010.000000      5.100000      4.64000
50%        29177.000000   2013.000000      7.050000      6.59000
75%        45081.000000   2016.000000      9.350000      8.88000
max        56045.000000   2018.000000     36.110000     38.24000

            MUNR_16    UNR_20to64   FUNR_20to64   MUNR_20to64  FLFPR_20to64  \
count  34567.000000  34567.000000  34567.000000  34567.000000  34567.000000
mean       7.860291      6.900073      6.482007      7.275457     70.086125
std        4.037657      3.446199      3.477956      3.990758      7.696499
min        0.000000      0.000000      0.000000      0.000000     33.600000
25%        5.200000      4.600000      4.200000      4.700000     65.100000
50%        7.390000      6.500000      6.000000      6.800000     70.600000
75%        9.920000      8.700000      8.250000      9.200000     75.500000
max       39.740000     33.900000     44.500000     45.500000    100.000000

       FLFPR_20to64_Under6  …   MFCCToddler  MFCCToddler_flag  \
count         34567.000000  …  23383.000000      23383.000000
mean             68.821409  …    106.759749          1.153359
std              11.758088  …     29.982431          0.532176
min               0.000000  …     43.080000          1.000000
25%              62.600000  …     85.085000          1.000000
50%              69.600000  …    100.250000          1.000000
75%              76.100000  …    124.950000          1.000000
max             100.000000  …    376.320000          3.000000

       MFCCPreschool  MFCCPreschool_flag  _75FCCInfant  _75FCCInfant_flag  \
count   23383.000000        23383.000000  23383.000000       23383.000000
mean      104.189510            1.287859    128.909289           1.792841
std        28.961701            0.696762     38.543010           0.818080
min        40.030000            1.000000     50.000000           1.000000
25%        84.255000            1.000000    100.830000           1.000000
```

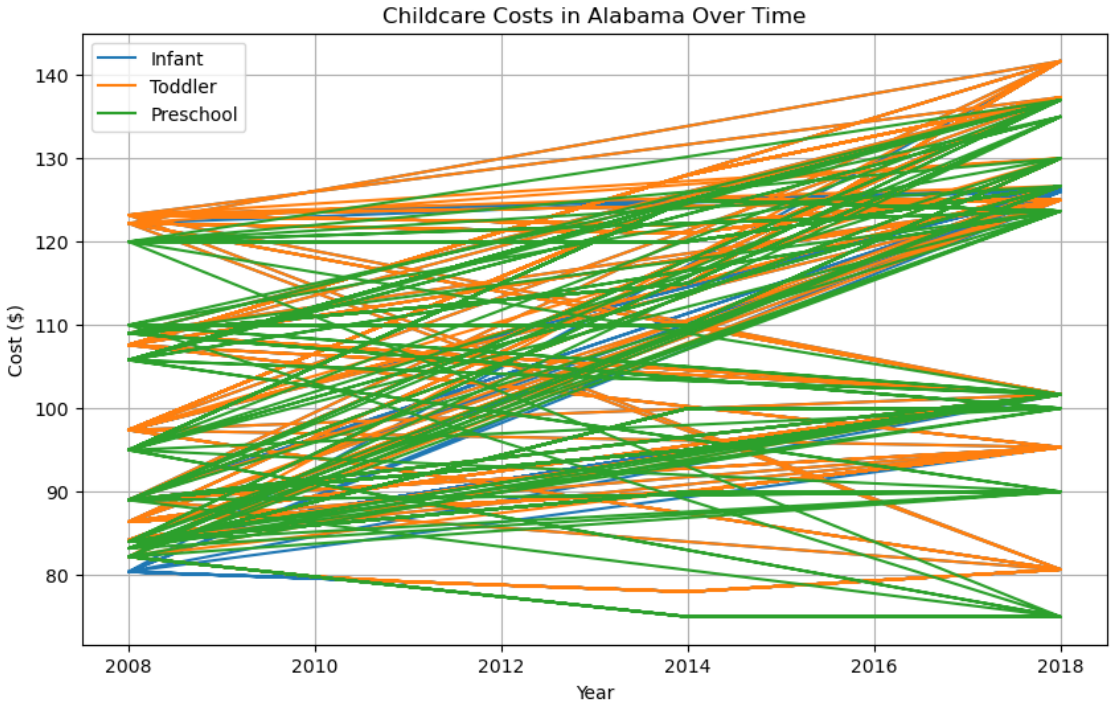
```
50%        99.650000            1.000000    123.150000           2.000000
75%       120.200000            1.000000    146.950000           3.000000
max       331.340000            3.000000    502.970000           3.000000

       _75FCCToddler  _75FCCToddler_flag  _75FCCPreschool  \
count   23383.000000         23383.00000     23383.000000
mean      120.784283             1.18800       117.897482
std        35.334666             0.58367        34.111188
min        50.000000             1.00000        46.450000
25%        95.850000             1.00000        95.000000
50%       115.000000             1.00000       112.500000
75%       136.270000             1.00000       132.760000
max       439.220000             3.00000       386.720000

       _75FCCPreschool_flag
count          23383.000000
mean               1.294316
std                0.708542
min                1.000000
25%                1.000000
50%                1.000000
75%                1.000000
max                3.000000

[8 rows x 224 columns]
Data types:
State_Name               object
State_Abbreviation       object
County_Name              object
County_FIPS_Code          int64
StudyYear                 int64
                         …
_75FCCInfant_flag       float64
_75FCCToddler           float64
_75FCCToddler_flag      float64
_75FCCPreschool         float64
_75FCCPreschool_flag    float64
Length: 227, dtype: object
```

[14]:
```python
# Define relevant columns
columns_of_interest = [
    'State_Name', 'County_Name', 'StudyYear',
    '_75FCCInfant', '_75FCCToddler', '_75FCCPreschool',
    '_75FCCInfant_flag', '_75FCCToddler_flag', '_75FCCPreschool_flag'
]

# Create a subset
```

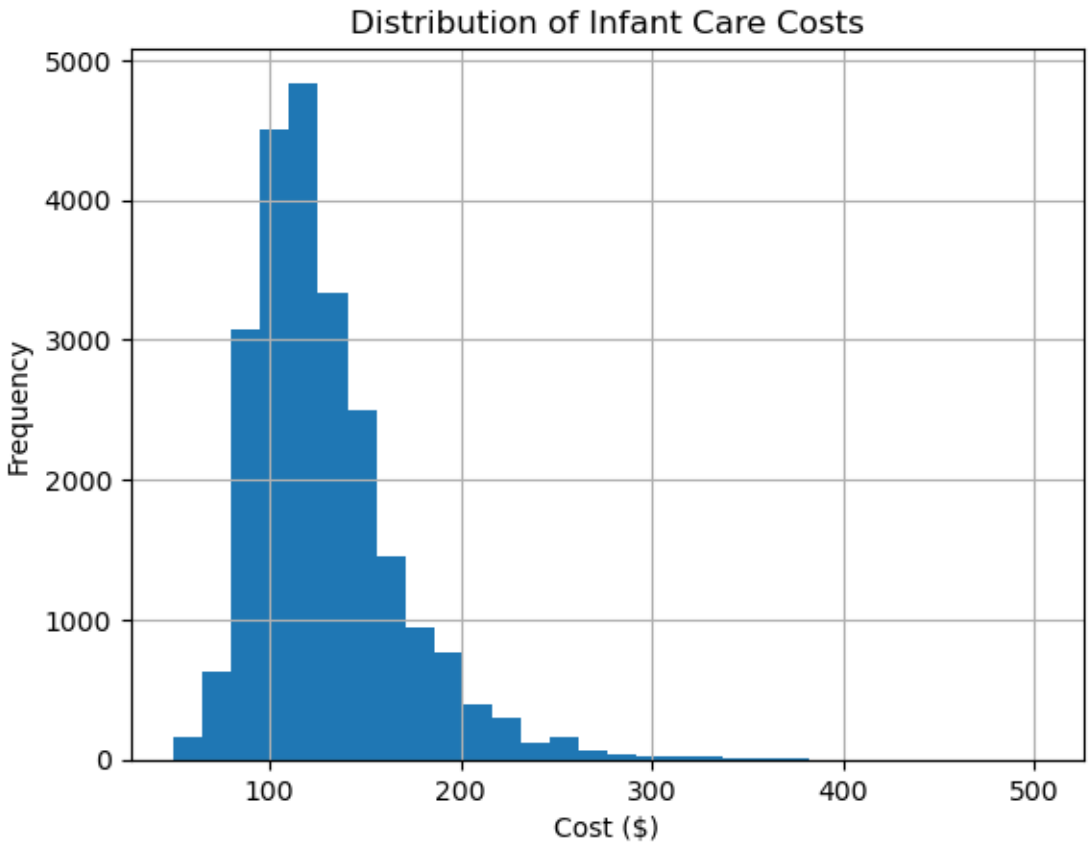
```
subset_data = data[columns_of_interest]

# Preview the subset
print(subset_data.head())
```

```
  State_Name     County_Name  StudyYear  _75FCCInfant  _75FCCToddler  \
0    Alabama  Autauga County       2008          97.4           97.4
1    Alabama  Autauga County       2009         102.0          102.0
2    Alabama  Autauga County       2010         106.6          106.6
3    Alabama  Autauga County       2011         111.2          111.2
4    Alabama  Autauga County       2012         115.8          115.8

   _75FCCPreschool  _75FCCInfant_flag  _75FCCToddler_flag  \
0             95.0                1.0                 3.0
1            100.0                1.0                 3.0
2            105.0                1.0                 3.0
3            110.0                1.0                 3.0
4            115.0                1.0                 3.0

   _75FCCPreschool_flag
0                   1.0
1                   1.0
2                   1.0
3                   1.0
4                   1.0
```

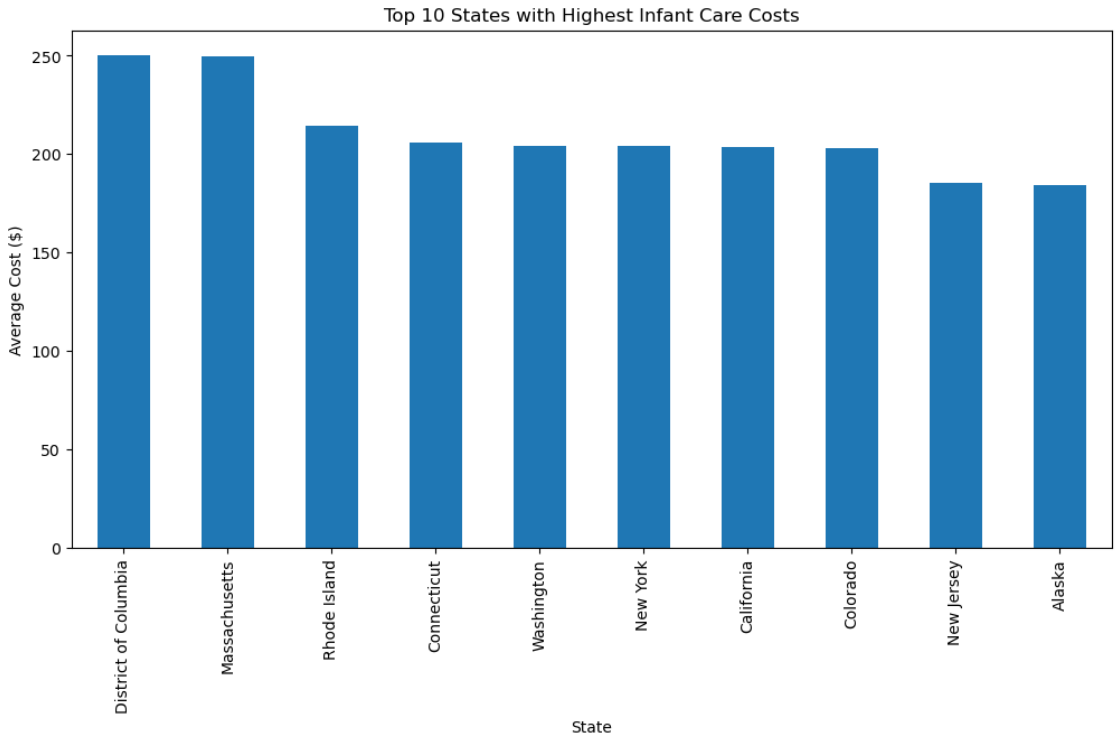
[18]:
```
import matplotlib.pyplot as plt

# Filter data for a specific state (e.g., Alabama)
state_data = subset_data[subset_data['State_Name'] == 'Alabama']

# Plot trends for infant, toddler, and preschool costs
plt.figure(figsize=(10, 6))
plt.plot(state_data['StudyYear'], state_data['_75FCCInfant'], label='Infant')
plt.plot(state_data['StudyYear'], state_data['_75FCCToddler'], label='Toddler')
plt.plot(state_data['StudyYear'], state_data['_75FCCPreschool'],␣
 ↪label='Preschool')
plt.title('Childcare Costs in Alabama Over Time')
plt.xlabel('Year')
plt.ylabel('Cost ($)')
plt.legend()
plt.grid(True)
plt.show()
```



[20]:
```
# Histogram of infant care costs
subset_data['_75FCCInfant'].hist(bins=30)
plt.title('Distribution of Infant Care Costs')
plt.xlabel('Cost ($)')
plt.ylabel('Frequency')
plt.show()
```



[22]:
```
# Group by state and calculate average costs
state_avg_costs = subset_data.groupby('State_Name')[['_75FCCInfant',␣
 ↪'_75FCCToddler', '_75FCCPreschool']].mean()

# Sort by infant care costs
state_avg_costs = state_avg_costs.sort_values(by='_75FCCInfant',␣
 ↪ascending=False)

# Plot the top 10 states for infant care costs
state_avg_costs.head(10)['_75FCCInfant'].plot(kind='bar', figsize=(12, 6))
plt.title('Top 10 States with Highest Infant Care Costs')
plt.xlabel('State')
plt.ylabel('Average Cost ($)')
plt.show()
```

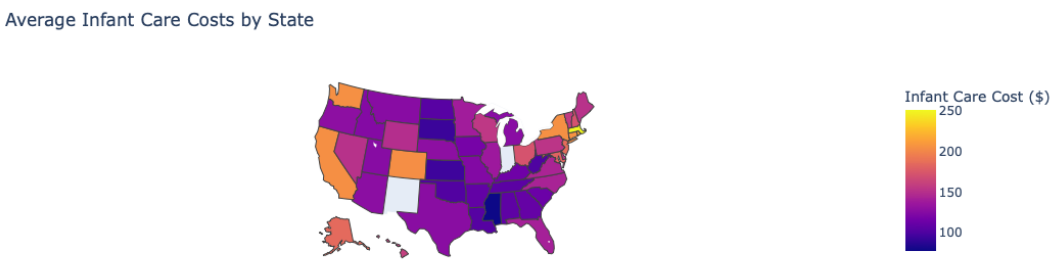


[47]:
```
# Check for missing data in the subset
missing_values = subset_data.isnull().sum()
print("Missing values per column:")
print(missing_values)

# Percentage of missing data
missing_percentage = (missing_values / len(subset_data)) * 100
print("\nPercentage of missing data:")
print(missing_percentage)
```

```
Missing values per column:
State_Name              0
County_Name             0
StudyYear               0
_75FCCInfant            0
_75FCCToddler           0
_75FCCPreschool         0
_75FCCInfant_flag       0
_75FCCToddler_flag      0
_75FCCPreschool_flag    0
dtype: int64

Percentage of missing data:
State_Name              0.0
```

```
County_Name             0.0
StudyYear               0.0
_75FCCInfant            0.0
_75FCCToddler           0.0
_75FCCPreschool         0.0
_75FCCInfant_flag       0.0
_75FCCToddler_flag      0.0
_75FCCPreschool_flag    0.0
dtype: float64
```

[49]:
```python
import plotly.express as px

# Calculate average infant care cost per state
state_avg = data.groupby("State_Abbreviation")["_75FCCInfant"].mean().
 ↪reset_index()

# Create a heatmap
fig = px.choropleth(
    state_avg,
    locations="State_Abbreviation",  # Use abbreviations for locations
    locationmode="USA-states",       # Focus on U.S. states
    color="_75FCCInfant",            # Column to determine the color intensity
    scope="usa",                     # Show U.S. states
    title="Average Infant Care Costs by State",
    labels={"_75FCCInfant": "Infant Care Cost ($)"}
)

# Display the map
fig.show()
```

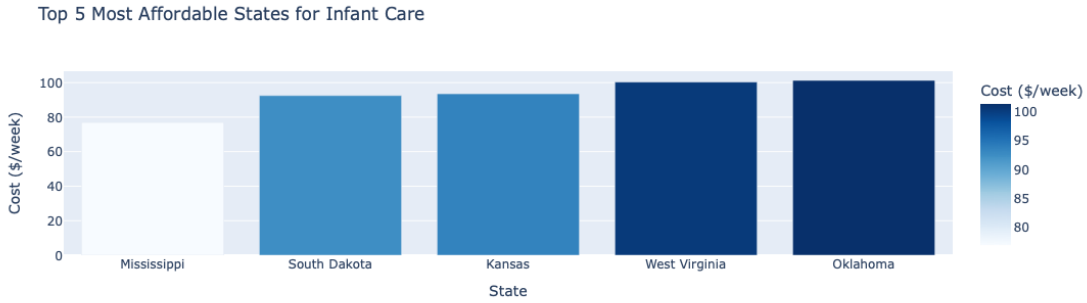


[51]:
```python
# Group by state and calculate average costs
state_avg = data.groupby("State_Name")["_75FCCInfant"].mean().reset_index()

# Find the top 5 most affordable and most expensive states
```

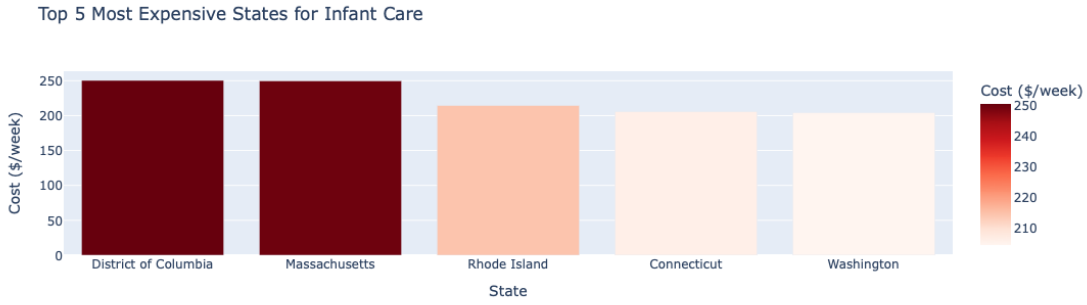
```
most_affordable = state_avg.nsmallest(5, "_75FCCInfant")
most_expensive = state_avg.nlargest(5, "_75FCCInfant")

print("Most Affordable States:")
print(most_affordable)

print("Most Expensive States:")
print(most_expensive)
```

```
Most Affordable States:
       State_Name  _75FCCInfant
24    Mississippi     76.909648
41   South Dakota     92.505611
16         Kansas     93.532087
48  West Virginia    100.329521
36       Oklahoma    101.275478
Most Expensive States:
              State_Name  _75FCCInfant
8   District of Columbia    250.400000
21         Massachusetts    249.749870
39          Rhode Island    214.432000
6            Connecticut    205.767045
47            Washington    204.249814
```

[53]:
```
import plotly.express as px

# Bar chart for most affordable states
fig_affordable = px.bar(
    most_affordable,
    x="State_Name",
    y="_75FCCInfant",
    title="Top 5 Most Affordable States for Infant Care",
    labels={"State_Name": "State", "_75FCCInfant": "Cost ($/week)"},
    color="_75FCCInfant",
    color_continuous_scale="Blues"
)

fig_affordable.show()
```



[55]:
```
# Bar chart for most expensive states
fig_expensive = px.bar(
    most_expensive,
    x="State_Name",
    y="_75FCCInfant",
    title="Top 5 Most Expensive States for Infant Care",
    labels={"State_Name": "State", "_75FCCInfant": "Cost ($/week)"},
    color="_75FCCInfant",
    color_continuous_scale="Reds"
)

fig_expensive.show()
```



[43]:
```
import dash
from dash import dcc, html
import pandas as pd
import plotly.express as px

# Load Data
```

```
file_path = "/Users/cheribeda/Desktop/nationaldatabaseofchildcareprices (1).
 ↪xlsx"
data = pd.read_excel(file_path)

# Prepare Data for Visualizations
state_avg = data.groupby("State_Abbreviation")["_75FCCInfant"].mean().
 ↪reset_index()

most_affordable = state_avg.nsmallest(5, "_75FCCInfant")
most_expensive = state_avg.nlargest(5, "_75FCCInfant")

# Create Heatmap
fig_heatmap = px.choropleth(
    state_avg,
    locations="State_Abbreviation",  # Use abbreviations for locations
    locationmode="USA-states",       # Focus on U.S. states
    color="_75FCCInfant",            # Column to determine the color intensity
    scope="usa",                     # Show U.S. states
    title="Average Infant Care Costs by State",
    labels={"_75FCCInfant": "Infant Care Cost ($)"}
)

# Create Bar Chart for Most Affordable States
fig_affordable = px.bar(
    most_affordable,
    x="State_Abbreviation",
    y="_75FCCInfant",
    title="Top 5 Most Affordable States for Infant Care",
    labels={"State_Abbreviation": "State", "_75FCCInfant": "Cost ($/week)"},
    color="_75FCCInfant",
    color_continuous_scale="Blues"
)

# Create Bar Chart for Most Expensive States
fig_expensive = px.bar(
    most_expensive,
    x="State_Abbreviation",
    y="_75FCCInfant",
    title="Top 5 Most Expensive States for Infant Care",
    labels={"State_Abbreviation": "State", "_75FCCInfant": "Cost ($/week)"},
    color="_75FCCInfant",
    color_continuous_scale="Reds"
)

# Initialize the Dash App
app = dash.Dash(__name__)
```

```
# Layout
app.layout = html.Div([
    html.H1("Childcare Costs Dashboard", style={'textAlign': 'center'}),

    # Heatmap
    html.Div([
        dcc.Graph(figure=fig_heatmap)  # Add heatmap to the dashboard
    ]),

    # Bar Charts
    html.Div([
        dcc.Graph(figure=fig_affordable),  # Add most affordable states bar␣
 ↪chart
        dcc.Graph(figure=fig_expensive)   # Add most expensive states bar chart
    ], style={'display': 'flex', 'justify-content': 'space-around'})  # Flexbox␣
 ↪layout for side-by-side charts
])

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

```
<IPython.lib.display.IFrame at 0x12577e7d0>
```

[ ]: