WeatherEDA_NYC

March 20, 2025

0.1 Data Viz: Weather EDA NYC

1 A: Sliding Scale: Weather Per Year:

To access, Please use the public link: https://colab.research.google.com/drive/1ptILKcYM5CCXevbIEx8Nu14Lj4V

[ ]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import ipywidgets as widgets
from IPython.display import display

df = pd.read_csv('weather.csv')

df['time'] = pd.to_datetime(df['time'])
df['year'] = df['time'].dt.year
df['month'] = df['time'].dt.month
df['Ftemp'] = (df['Ktemp'] - 273.15) * (9/5) + 32

def plot(year):
    filtered_data = df[df['year'] == year]
    avg_monthly_temp = filtered_data.groupby('month')['Ftemp'].mean()

    plt.figure(figsize=(10, 5))
    sns.lineplot(x=avg_monthly_temp.index, y=avg_monthly_temp.values,
 ↪marker='o')

    plt.xlabel("Month")
    plt.ylabel("Average Temperature (°F)")
    plt.title(f"Average Monthly Temperature for {year}")
    plt.xticks(
        ticks=range(1, 13),
        labels=['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun',
                'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec']
    )
    plt.grid()
    plt.show()
```

```python
def update(change):
    with output_widget:
        output_widget.clear_output(wait=True)
        plot(year_slider.value)

output_widget = widgets.Output()
year_slider = widgets.IntSlider(
    value=df['year'].min(),
    min=df['year'].min(),
    max=df['year'].max(),
    step=1,
    description="Select Year:"
)

year_slider.observe(update, names='value')
display(year_slider, output_widget)
```

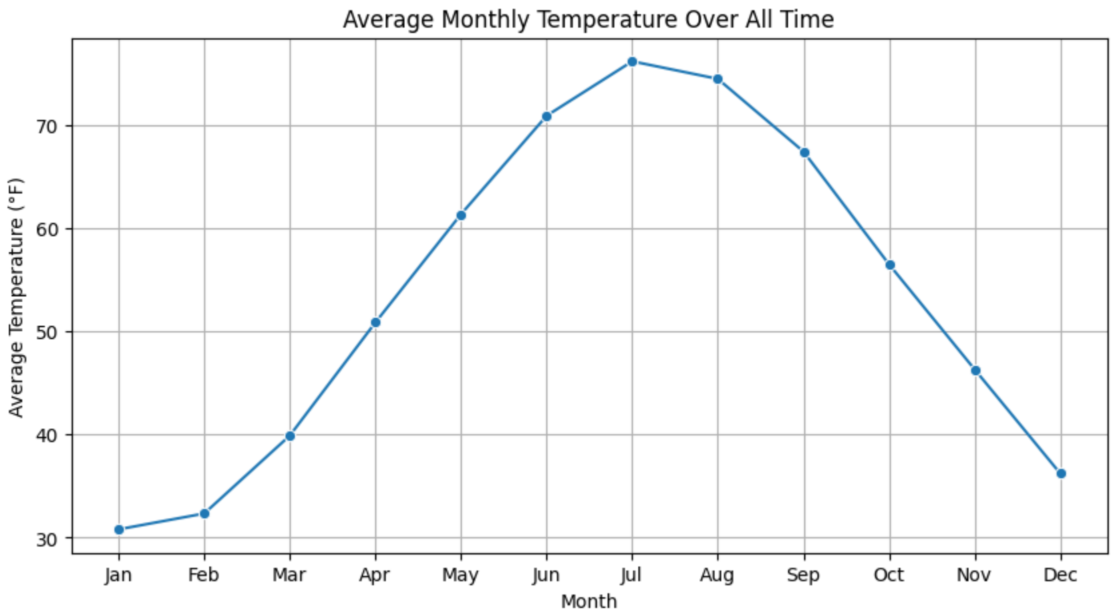
```
IntSlider(value=1950, description='Select Year:', max=2021, min=1950)
```

```
Output()
```

1.1 Including all time monthly avg. temp

[ ]:

```python
monthly_avg_all_time = df.groupby('month')['Ftemp'].mean()

plt.figure(figsize=(10, 5))
sns.lineplot(x=monthly_avg_all_time.index, y=monthly_avg_all_time.values,␣
 ↪marker='o')
plt.xlabel("Month")
plt.ylabel("Average Temperature (°F)")
plt.title("Average Monthly Temperature Over All Time")
plt.xticks(range(1, 13),
           ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep',␣
 ↪'Oct', 'Nov', 'Dec'])
plt.grid()
plt.show()
```



Monthy average temperatures follow a sinusoidal oscillation in which temperatures plunge into the winter and peak in the middle of the summer (in New York City).

2 B: Finding first year avg. temp passes 55 degrees

[ ]:
```
year = df.groupby('year')['Ftemp'].mean().reset_index()
pd.set_option('display.max_rows', None)
print(year)
```

```
    year      Ftemp
0   1950  52.776070
1   1951  53.822162
2   1952  54.372742
3   1953  55.295208
4   1954  53.463150
5   1955  53.596382
6   1956  52.687344
7   1957  54.043501
8   1958  51.603038
9   1959  54.059413
10  1960  51.990128
11  1961  52.206270
12  1962  51.400343
13  1963  51.784289
14  1964  52.622873
15  1965  52.075238
```

16	1966	52.663244
17	1967	51.378613
18	1968	52.516600
19	1969	52.835949
20	1970	52.762484
21	1971	53.563340
22	1972	52.550571
23	1973	54.595329
24	1974	53.544314
25	1975	54.455995
26	1976	52.378620
27	1977	52.999369
28	1978	51.512315
29	1979	53.171792
30	1980	52.773591
31	1981	52.989914
32	1982	52.866569
33	1983	53.525039
34	1984	53.464880
35	1985	53.859955
36	1986	53.593089
37	1987	53.580625
38	1988	53.003395
39	1989	52.962079
40	1990	55.598589
41	1991	55.555546
42	1992	52.545168
43	1993	53.981405
44	1994	53.874545
45	1995	54.101552
46	1996	52.664015
47	1997	53.387761
48	1998	56.114906
49	1999	54.955241
50	2000	52.762132
51	2001	54.788883
52	2002	55.259167
53	2003	52.436900
54	2004	53.453158
55	2005	54.055888
56	2006	55.748507
57	2007	54.272553
58	2008	54.459235
59	2009	53.355559
60	2010	55.695993
61	2011	55.524808
62	2012	56.597076
63	2013	53.899689

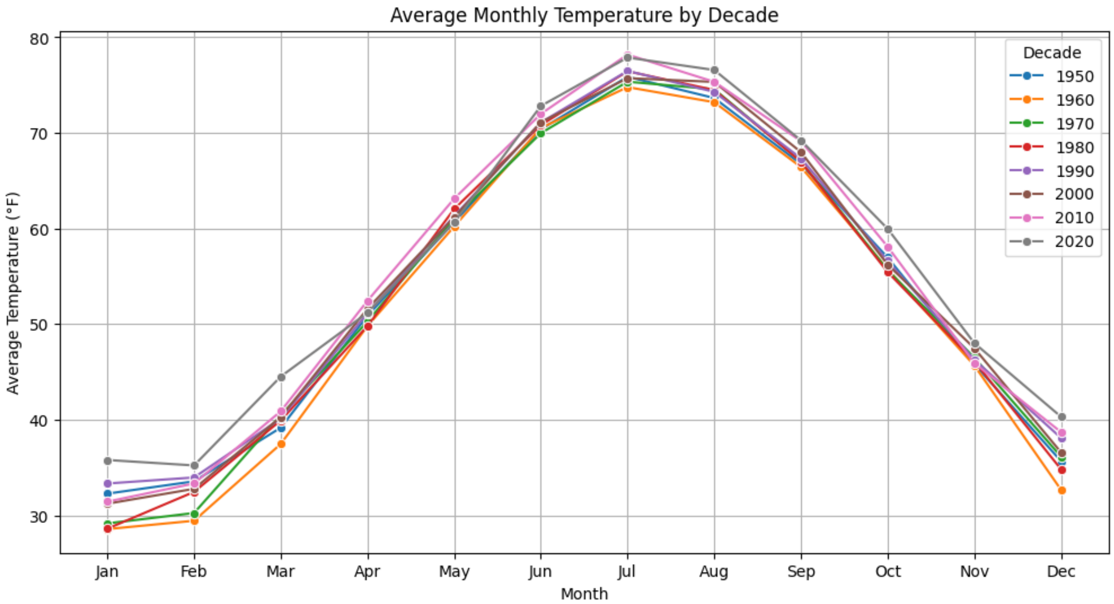
64	2014	52.617706
65	2015	54.554550
66	2016	55.862648
67	2017	55.456930
68	2018	54.944623
69	2019	54.809348
70	2020	56.145076
71	2021	56.078091

3 First year surpassing 55 degrees is 1953, with an average temp of 55.295208 degrees Farenheit

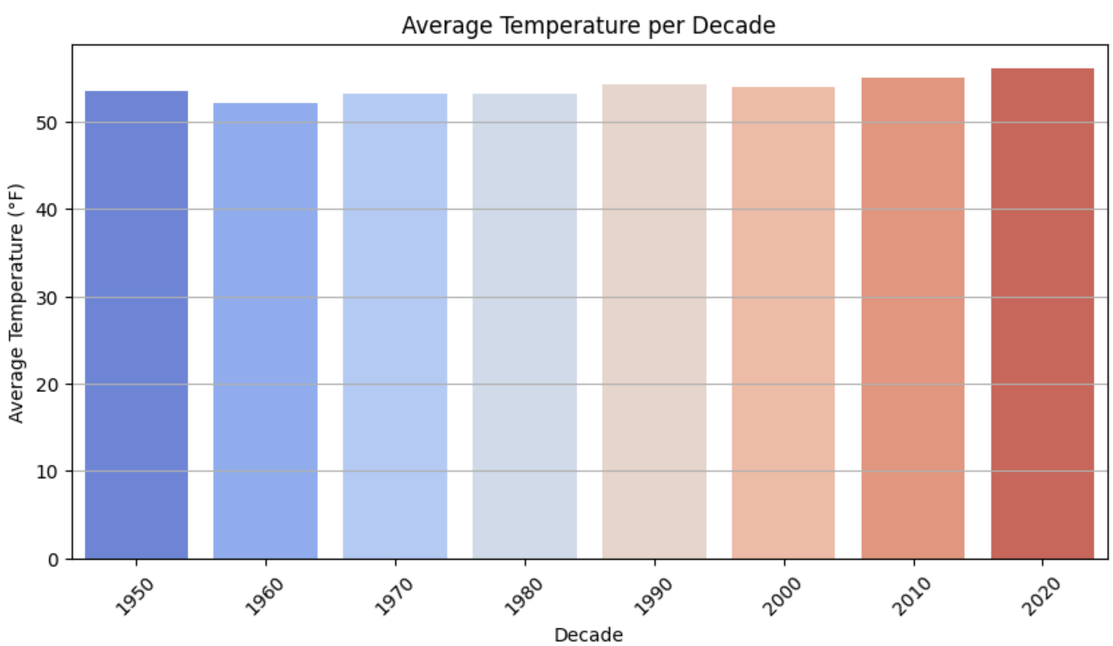
4 C: Looking into Temperature Increases From 1950 to 2021

4.1 Avg. Monthy Temperature By Decade Plotted + Avg. Temperature By Decade Plotted

[ ]:
```
df['decade'] = (df['year'] // 10) * 10



plt.figure(figsize=(12, 6))
sns.lineplot(
    data=df.groupby(['decade', 'month'])['Ftemp'].mean().reset_index(),
    x='month', y='Ftemp', hue='decade', palette='tab10', marker='o'
)
plt.xlabel("Month")
plt.ylabel("Average Temperature (°F)")
plt.title("Average Monthly Temperature by Decade")
plt.xticks(range(1, 13),
           ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep',␣
 ↪'Oct', 'Nov', 'Dec'])
plt.legend(title="Decade")
plt.grid()
plt.show()

plt.figure(figsize=(10, 5))
sns.barplot(
    data=df.groupby('decade')['Ftemp'].mean().reset_index(),
    x='decade', y='Ftemp', palette='coolwarm'
)
plt.xlabel("Decade")
plt.ylabel("Average Temperature (°F)")
plt.title("Average Temperature per Decade")
plt.xticks(rotation=45)
plt.grid(axis='y')
plt.show()
```




```
<ipython-input-45-be40e402a657>:20: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same
effect.

  sns.barplot(
```

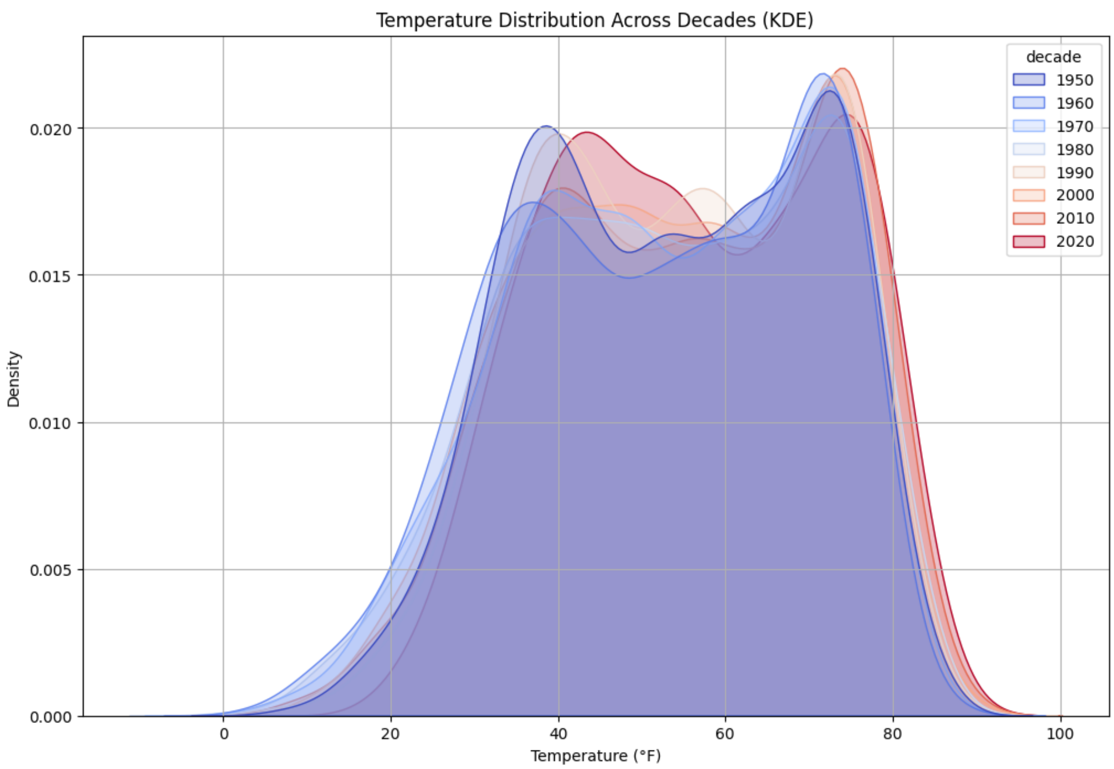


From the top plot, we see that recent decades tend to be the warmest decades across the months and seasons.

From the bottom plot, it is evident that there is a slight and steady increase in average temperature over the past 7 decades.

4.2 Kernel Density Estimate Plot For Temperature By Decade

[ ]:
```
df_sorted = df.sort_values(by="decade")

plt.figure(figsize=(12, 8))
sns.kdeplot(
    data=df_sorted, x="Ftemp", hue="decade", fill=True, common_norm=False,␣
  ↪palette="coolwarm"
)
plt.xlabel("Temperature (°F)")
plt.ylabel("Density")
plt.title("Temperature Distribution Across Decades (KDE)")
plt.grid()
plt.show()
```



Here, it seems that we see a drift towards warmer temperatures overall throughout the decades (with higher minimums and higher maximums).

4.3 Temperature Changes over the decades (quantified)

[ ]:
```
decade_avg = df.groupby('decade')['Ftemp'].mean().reset_index()
decade_avg['Temp_Change'] = decade_avg['Ftemp'].diff()
decade_avg['Rate_of_Change'] = decade_avg['Temp_Change'] / 10

print("Overall temperature increase (first to last decade):",
      decade_avg['Ftemp'].iloc[-1] - decade_avg['Ftemp'].iloc[0])
print("Average temperature delta per decade:",
      decade_avg['Temp_Change'].mean())
print("Max temperature increase in a decade:",
      decade_avg['Temp_Change'].max())
print("Min temperature increase in a decade:",
      decade_avg['Temp_Change'].min())
```

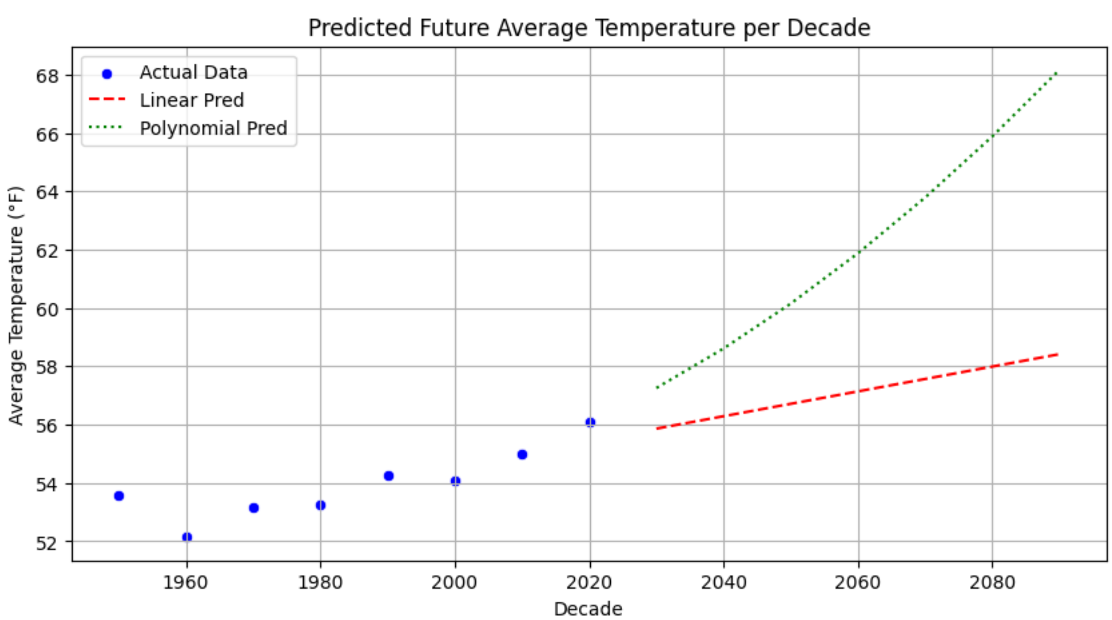
```
Overall temperature increase (first to last decade): 2.5397512004770704
Average temperature delta per decade: 0.3628216000681529
Max temperature increase in a decade: 1.1146167311451904
Min temperature increase in a decade: -1.4243353222243087
```

4.4 Predicting Future Decade Average Temperature

[ ]:
```
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures

X = decade_avg[['decade']]
y = decade_avg['Ftemp']

lin_reg = LinearRegression()
lin_reg.fit(X, y)

future_decades = np.array(range(decade_avg['decade'].max() + 10, 2100, 10)).
 ↪reshape(-1, 1)
future_predictions = lin_reg.predict(future_decades)

poly = PolynomialFeatures(degree=2)
X_poly = poly.fit_transform(X)

poly_reg = LinearRegression()
poly_reg.fit(X_poly, y)

future_X_poly = poly.transform(future_decades)
future_poly_predictions = poly_reg.predict(future_X_poly)

plt.figure(figsize=(10, 5))
```

```python
# Plot actual data
sns.scatterplot(x=decade_avg['decade'], y=decade_avg['Ftemp'], label="Actual␣
↪Data", color='blue')
plt.plot(future_decades, future_predictions, label="Linear Pred",␣
↪linestyle='dashed', color='red')
plt.plot(future_decades, future_poly_predictions, label="Polynomial Pred",␣
↪linestyle='dotted', color='green')

# Formatting
plt.xlabel("Decade")
plt.ylabel("Average Temperature (°F)")
plt.title("Predicted Future Average Temperature per Decade")
plt.legend()
plt.grid()

plt.show()

print("Linear Regression R² Score:", lin_reg.score(X, y))
print("Polynomial Regression R² Score:", poly_reg.score(X_poly, y))
```

```
/usr/local/lib/python3.11/dist-packages/sklearn/utils/validation.py:2739:
UserWarning: X does not have valid feature names, but LinearRegression was
fitted with feature names
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/sklearn/utils/validation.py:2739:
UserWarning: X does not have valid feature names, but PolynomialFeatures was
fitted with feature names
  warnings.warn(
```



```
Linear Regression R² Score: 0.7326524788744002
Polynomial Regression R² Score: 0.87209698685391
```

4.5 Results:

Given the higher R^2 value for the polynomial regression model compared to the linear regression model, it suggests that a ***non-linear trend*** better fits the historical temperature data. This aligns with the rise in concern about climate change.