Project 4

Name: Jash Dedhia

M-ID: M15047151 (dedhiaja)

Date: November 03, 2024

Github Link: https://github.com/dedhiaja/Project-4

1.

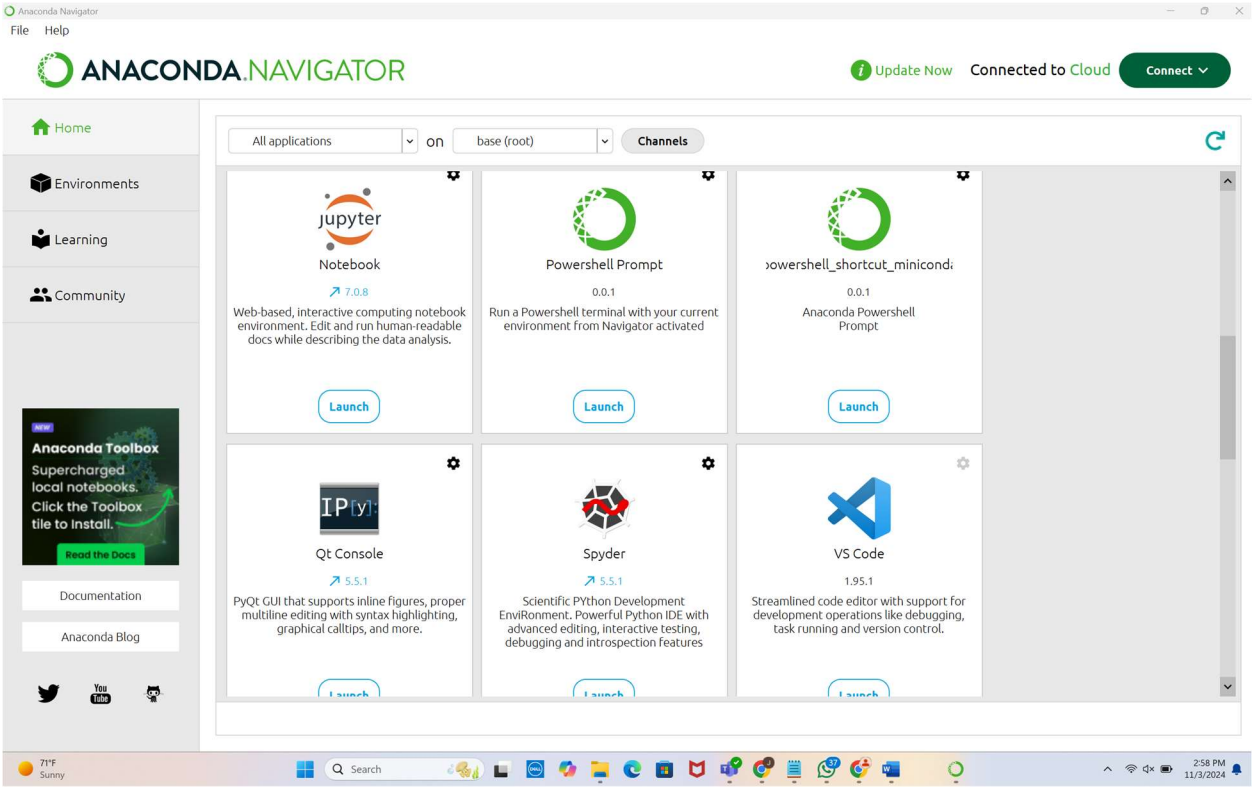
```
In [1]: # Name: Jash Dedhia
        # Date: 2nd Nov, 2024
        # Project 4
```

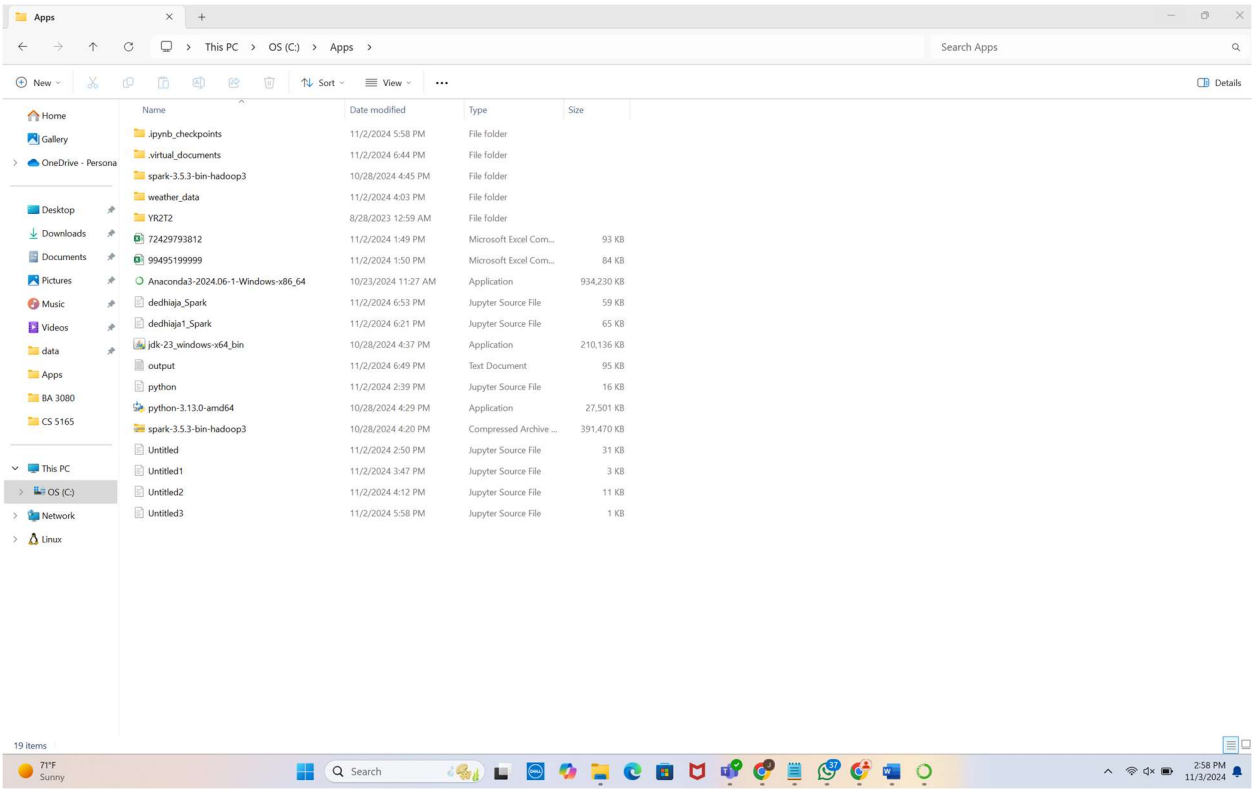
```
In [3]: !pip install pyspark
```

```
Requirement already satisfied: pyspark in c:\users\dedhi\anaconda3\lib\site-packages (3.5.3)
Requirement already satisfied: py4j==0.10.9.7 in c:\users\dedhi\anaconda3\lib\site-packages (from pyspark) (0.10.9.7)
```

```
In [5]: !pip install requests beautifulsoup4
```

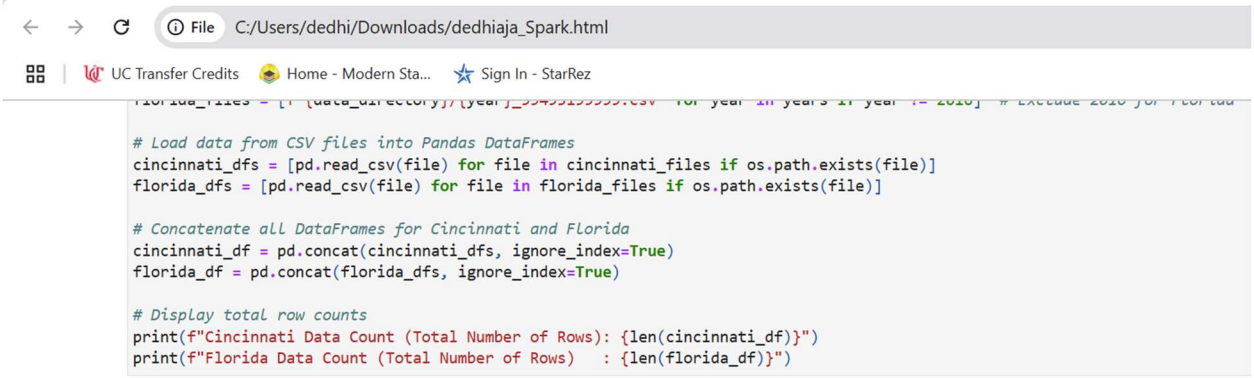
```
Requirement already satisfied: requests in c:\users\dedhi\anaconda3\lib\site-packages (2.32.2)
Requirement already satisfied: beautifulsoup4 in c:\users\dedhi\anaconda3\lib\site-packages (4.12.3)
Requirement already satisfied: charset-normalizer<4,>=2 in c:\users\dedhi\anaconda3\lib\site-packages (from requests) (2.0.4)
Requirement already satisfied: idna<4,>=2.5 in c:\users\dedhi\anaconda3\lib\site-packages (from requests) (3.7)
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\users\dedhi\anaconda3\lib\site-packages (from requests) (2.2.2)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\dedhi\anaconda3\lib\site-packages (from requests) (2024.8.30)
Requirement already satisfied: soupsieve>1.2 in c:\users\dedhi\anaconda3\lib\site-packages (from beautifulsoup4) (2.5)
```





2.

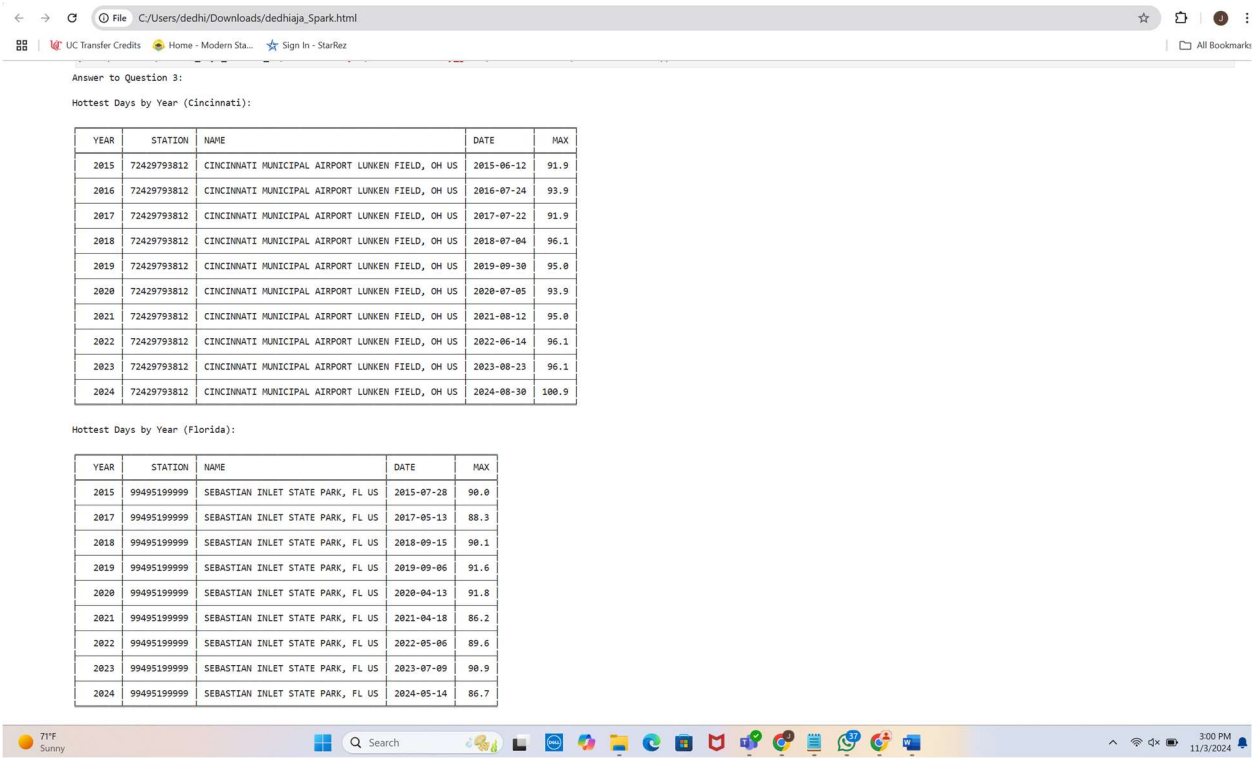
```
Answer to Question 2:
Downloaded: ./weather_data\2015_72429793812.csv
Downloaded: ./weather_data\2015_99495199999.csv
Downloaded: ./weather_data\2016_72429793812.csv
Downloaded: ./weather_data\2017_72429793812.csv
Downloaded: ./weather_data\2017_99495199999.csv
Downloaded: ./weather_data\2018_72429793812.csv
Downloaded: ./weather_data\2018_99495199999.csv
Downloaded: ./weather_data\2019_72429793812.csv
Downloaded: ./weather_data\2019_99495199999.csv
Downloaded: ./weather_data\2020_72429793812.csv
Downloaded: ./weather_data\2020_99495199999.csv
Downloaded: ./weather_data\2021_72429793812.csv
Downloaded: ./weather_data\2021_99495199999.csv
Downloaded: ./weather_data\2022_72429793812.csv
Downloaded: ./weather_data\2022_99495199999.csv
Downloaded: ./weather_data\2023_72429793812.csv
Downloaded: ./weather_data\2023_99495199999.csv
Downloaded: ./weather_data\2024_72429793812.csv
Downloaded: ./weather_data\2024_99495199999.csv
Cincinnati --> Year: 2015, Station: 72429793812, Count: 365
Florida --> Year: 2015, Station: 99495199999, Count: 355
Cincinnati --> Year: 2016, Station: 72429793812, Count: 366
Cincinnati --> Year: 2017, Station: 72429793812, Count: 365
Florida --> Year: 2017, Station: 99495199999, Count: 283
Cincinnati --> Year: 2018, Station: 72429793812, Count: 365
Florida --> Year: 2018, Station: 99495199999, Count: 363
Cincinnati --> Year: 2019, Station: 72429793812, Count: 365
Florida --> Year: 2019, Station: 99495199999, Count: 345
Cincinnati --> Year: 2020, Station: 72429793812, Count: 366
Florida --> Year: 2020, Station: 99495199999, Count: 365
Cincinnati --> Year: 2021, Station: 72429793812, Count: 365
Florida --> Year: 2021, Station: 99495199999, Count: 104
Cincinnati --> Year: 2022, Station: 72429793812, Count: 365
Florida --> Year: 2022, Station: 99495199999, Count: 259
Cincinnati --> Year: 2023, Station: 72429793812, Count: 365
Florida --> Year: 2023, Station: 99495199999, Count: 276
Cincinnati --> Year: 2024, Station: 72429793812, Count: 301
Florida --> Year: 2024, Station: 99495199999, Count: 133

Total Results: 19 (as expected)
```




```
# Load data from CSV files into Pandas DataFrames
cincinnati_dfs = [pd.read_csv(file) for file in cincinnati_files if os.path.exists(file)]
florida_dfs = [pd.read_csv(file) for file in florida_files if os.path.exists(file)]

# Concatenate all DataFrames for Cincinnati and Florida
cincinnati_df = pd.concat(cincinnati_dfs, ignore_index=True)
florida_df = pd.concat(florida_dfs, ignore_index=True)

# Display total row counts
print(f"Cincinnati Data Count (Total Number of Rows): {len(cincinnati_df)}")
print(f"Florida Data Count (Total Number of Rows)   : {len(florida_df)}")
```

```
Cincinnati Data Count (Total Number of Rows): 3588
Florida Data Count (Total Number of Rows)   : 2483
```

3.



Answer to Question 3:

Hottest Days by Year (Cincinnati):

YEAR	STATION	NAME	DATE	MAX
2015	72429793812	CINCINNATI MUNICIPAL AIRPORT LUNKEN FIELD, OH US	2015-06-12	91.9
2016	72429793812	CINCINNATI MUNICIPAL AIRPORT LUNKEN FIELD, OH US	2016-07-24	93.9
2017	72429793812	CINCINNATI MUNICIPAL AIRPORT LUNKEN FIELD, OH US	2017-07-22	91.9
2018	72429793812	CINCINNATI MUNICIPAL AIRPORT LUNKEN FIELD, OH US	2018-07-04	96.1
2019	72429793812	CINCINNATI MUNICIPAL AIRPORT LUNKEN FIELD, OH US	2019-09-30	95.0
2020	72429793812	CINCINNATI MUNICIPAL AIRPORT LUNKEN FIELD, OH US	2020-07-05	93.9
2021	72429793812	CINCINNATI MUNICIPAL AIRPORT LUNKEN FIELD, OH US	2021-08-12	95.0
2022	72429793812	CINCINNATI MUNICIPAL AIRPORT LUNKEN FIELD, OH US	2022-06-14	96.1
2023	72429793812	CINCINNATI MUNICIPAL AIRPORT LUNKEN FIELD, OH US	2023-08-23	96.1
2024	72429793812	CINCINNATI MUNICIPAL AIRPORT LUNKEN FIELD, OH US	2024-08-30	100.9

Hottest Days by Year (Florida):

YEAR	STATION	NAME	DATE	MAX
2015	99495199999	SEBASTIAN INLET STATE PARK, FL US	2015-07-28	90.0
2017	99495199999	SEBASTIAN INLET STATE PARK, FL US	2017-05-13	88.3
2018	99495199999	SEBASTIAN INLET STATE PARK, FL US	2018-09-15	90.1
2019	99495199999	SEBASTIAN INLET STATE PARK, FL US	2019-09-06	91.6
2020	99495199999	SEBASTIAN INLET STATE PARK, FL US	2020-04-13	91.8
2021	99495199999	SEBASTIAN INLET STATE PARK, FL US	2021-04-18	86.2
2022	99495199999	SEBASTIAN INLET STATE PARK, FL US	2022-05-06	89.6
2023	99495199999	SEBASTIAN INLET STATE PARK, FL US	2023-07-09	90.9
2024	99495199999	SEBASTIAN INLET STATE PARK, FL US	2024-05-14	86.7

Overall Hottest Day by Year (Cincinnati and Florida):

YEAR	STATION	NAME	DATE	MAX
2015	72429793812	CINCINNATI MUNICIPAL AIRPORT LUNKEN FIELD, OH US	2015-06-12	91.9
2016	72429793812	CINCINNATI MUNICIPAL AIRPORT LUNKEN FIELD, OH US	2016-07-24	93.9
2017	72429793812	CINCINNATI MUNICIPAL AIRPORT LUNKEN FIELD, OH US	2017-07-22	91.9
2018	72429793812	CINCINNATI MUNICIPAL AIRPORT LUNKEN FIELD, OH US	2018-07-04	96.1
2019	72429793812	CINCINNATI MUNICIPAL AIRPORT LUNKEN FIELD, OH US	2019-09-30	95.0
2020	72429793812	CINCINNATI MUNICIPAL AIRPORT LUNKEN FIELD, OH US	2020-07-05	93.9
2021	72429793812	CINCINNATI MUNICIPAL AIRPORT LUNKEN FIELD, OH US	2021-08-12	95.0
2022	72429793812	CINCINNATI MUNICIPAL AIRPORT LUNKEN FIELD, OH US	2022-06-14	96.1
2023	72429793812	CINCINNATI MUNICIPAL AIRPORT LUNKEN FIELD, OH US	2023-08-23	96.1
2024	72429793812	CINCINNATI MUNICIPAL AIRPORT LUNKEN FIELD, OH US	2024-08-30	100.9

4.

Answer to Question 4:

Coldest Day Overall in March (2015-2024) across Cincinnati and Florida:

Year	Station ID	Station Name	Date	Min Temp (°F)
2015	72429793812	CINCINNATI MUNICIPAL AIRPORT LUNKEN FIELD, OH US	2015-03-06	3.2

5.

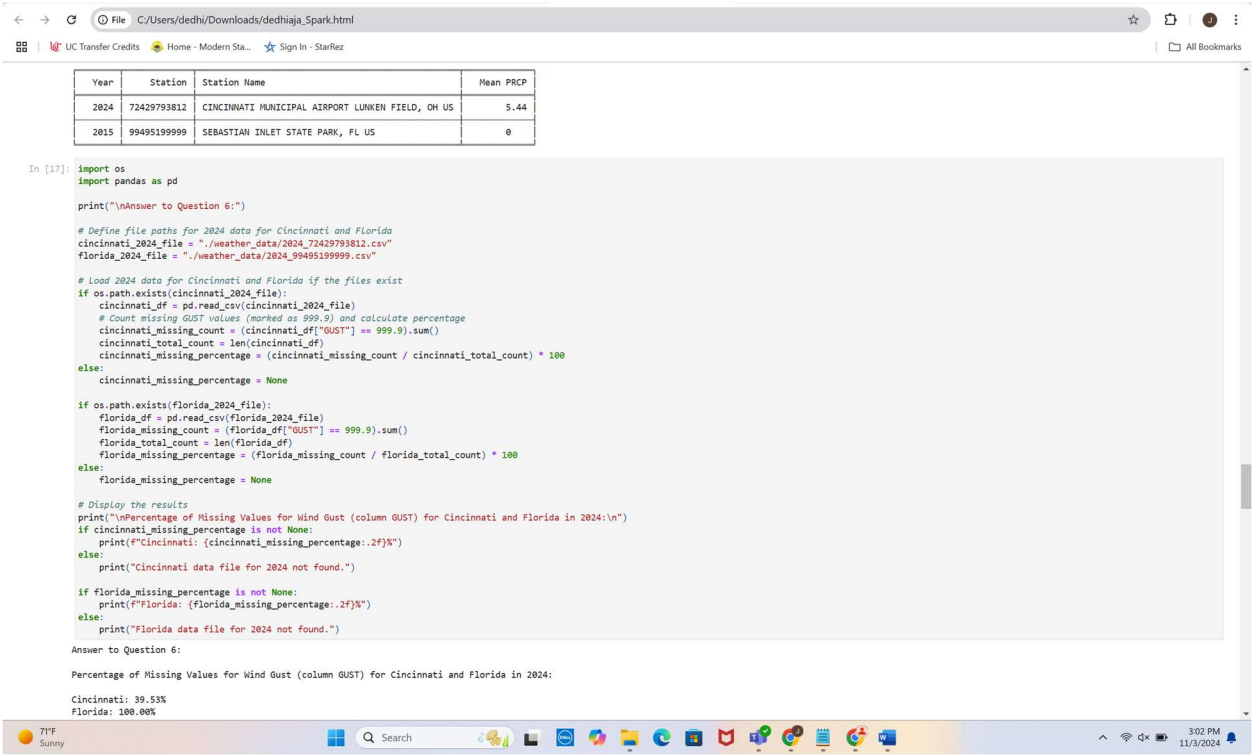

```python
# Calculate mean precipitation by year for Florida (excluding 2016 as data is unavailable)
florida_precip_data = []

for year in years:
    if year == 2016:
        continue
    file_path = f"{data_directory}/{year}_99495199999.csv"
    if os.path.exists(file_path):
        df = pd.read_csv(file_path)
        df = df[df["PRCP"] != 9999.9]
        if not df.empty:
            mean_prcp = df["PRCP"].mean()
            florida_precip_data.append({
                "YEAR": year,
                "STATION": str(df["STATION"].iloc[0]),  # Convert to string to preserve full ID
                "NAME": df["NAME"].iloc[0],
                "Mean_PRCP": mean_prcp
            })

florida_precip_df = pd.DataFrame(florida_precip_data)
florida_result = florida_precip_df.loc[florida_precip_df["Mean_PRCP"].idxmax()]

# Prepare data for display
results = [
    {
        "Year": int(cincinnati_result["YEAR"]),
        "Station": cincinnati_result["STATION"],
        "Station Name": cincinnati_result["NAME"],
        "Mean PRCP": round(cincinnati_result["Mean_PRCP"], 2)
    },
    {
        "Year": int(florida_result["YEAR"]),
        "Station": florida_result["STATION"],
        "Station Name": florida_result["NAME"],
        "Mean PRCP": round(florida_result["Mean_PRCP"], 2)
    }
]

# Display the results in a well-formatted table
print("\nYear with Most Precipitation for Cincinnati and Florida:\n")
print(tabulate(results, headers="keys", tablefmt="fancy_grid"))
```

Answer to Question 5:

Year with Most Precipitation for Cincinnati and Florida:

Year	Station	Station Name	Mean PRCP
2024	72429793812	CINCINNATI MUNICIPAL AIRPORT LUNKEN FIELD, OH US	5.44
2015	99495199999	SEBASTIAN INLET STATE PARK, FL US	0

6.

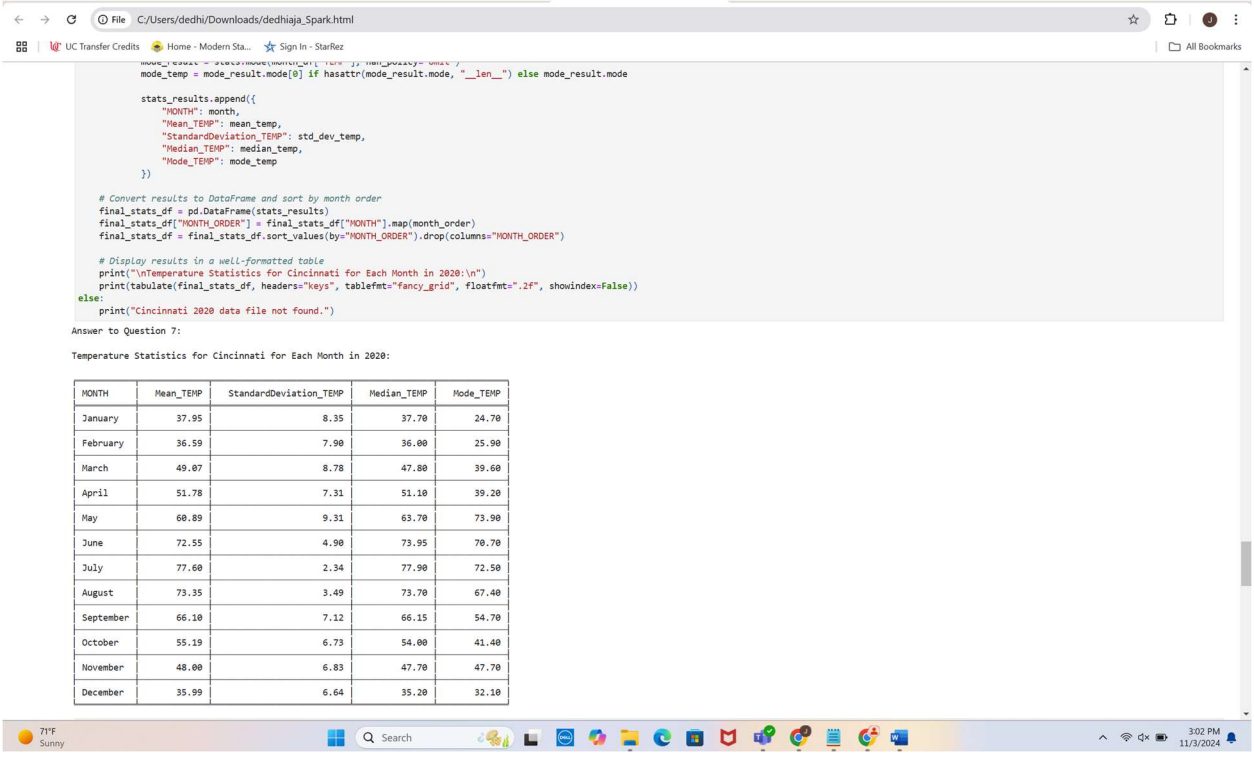


Year	Station	Station Name	Mean PRCP
2024	72429793812	CINCINNATI MUNICIPAL AIRPORT LUNKEN FIELD, OH US	5.44
2015	99495199999	SEBASTIAN INLET STATE PARK, FL US	0

```
In [17]: import os
         import pandas as pd

         print("\nAnswer to Question 6:")

         # Define file paths for 2024 data for Cincinnati and Florida
         cincinnati_2024_file = "./weather_data/2024_72429793812.csv"
         florida_2024_file = "./weather_data/2024_99495199999.csv"

         # Load 2024 data for Cincinnati and Florida if the files exist
         if os.path.exists(cincinnati_2024_file):
             cincinnati_df = pd.read_csv(cincinnati_2024_file)
             # Count missing GUST values (marked as 999.9) and calculate percentage
             cincinnati_missing_count = (cincinnati_df["GUST"] == 999.9).sum()
             cincinnati_total_count = len(cincinnati_df)
             cincinnati_missing_percentage = (cincinnati_missing_count / cincinnati_total_count) * 100
         else:
             cincinnati_missing_percentage = None

         if os.path.exists(florida_2024_file):
             florida_df = pd.read_csv(florida_2024_file)
             florida_missing_count = (florida_df["GUST"] == 999.9).sum()
             florida_total_count = len(florida_df)
             florida_missing_percentage = (florida_missing_count / florida_total_count) * 100
         else:
             florida_missing_percentage = None

         # Display the results
         print("\nPercentage of Missing Values for Wind Gust (column GUST) for Cincinnati and Florida in 2024:\n")
         if cincinnati_missing_percentage is not None:
             print(f"Cincinnati: {cincinnati_missing_percentage:.2f}%")
         else:
             print("Cincinnati data file for 2024 not found.")

         if florida_missing_percentage is not None:
             print(f"Florida: {florida_missing_percentage:.2f}%")
         else:
             print("Florida data file for 2024 not found.")
```

```
Answer to Question 6:

Percentage of Missing Values for Wind Gust (column GUST) for Cincinnati and Florida in 2024:

Cincinnati: 39.53%
Florida: 100.00%
```

7.



```
             mode_temp = mode_result.mode[0] if hasattr(mode_result.mode, "__len__") else mode_result.mode

             stats_results.append({
                 "MONTH": month,
                 "Mean_TEMP": mean_temp,
                 "StandardDeviation_TEMP": std_dev_temp,
                 "Median_TEMP": median_temp,
                 "Mode_TEMP": mode_temp
             })

     # Convert results to DataFrame and sort by month order
     final_stats_df = pd.DataFrame(stats_results)
     final_stats_df["MONTH_ORDER"] = final_stats_df["MONTH"].map(month_order)
     final_stats_df = final_stats_df.sort_values(by="MONTH_ORDER").drop(columns="MONTH_ORDER")

     # Display results in a well-formatted table
     print("\nTemperature Statistics for Cincinnati for Each Month in 2020:\n")
     print(tabulate(final_stats_df, headers="keys", tablefmt="fancy_grid", floatfmt=".2f", showindex=False))
 else:
     print("Cincinnati 2020 data file not found.")
```

```
Answer to Question 7:

Temperature Statistics for Cincinnati for Each Month in 2020:
```

MONTH	Mean_TEMP	StandardDeviation_TEMP	Median_TEMP	Mode_TEMP
January	37.95	8.35	37.70	24.70
February	36.59	7.90	36.00	25.90
March	49.07	8.78	47.80	39.60
April	51.78	7.31	51.10	39.20
May	60.89	9.31	63.70	73.90
June	72.55	4.90	73.95	70.70
July	77.60	2.34	77.90	72.50
August	73.35	3.49	73.70	67.40
September	66.10	7.12	66.15	54.70
October	55.19	6.73	54.00	41.40
November	48.00	6.83	47.70	47.70
December	35.99	6.64	35.20	32.10

8.

Answer to Question 8:

Top 10 Days with the Lowest Wind Chill for Cincinnati in 2017:

NAME	DATE	TEMP	WDSP	Wind_Chill
CINCINNATI MUNICIPAL AIRPORT LUNKEN FIELD, OH US	2017-01-07	10.50	7.00	-0.41
CINCINNATI MUNICIPAL AIRPORT LUNKEN FIELD, OH US	2017-12-31	11.00	5.30	2.03
CINCINNATI MUNICIPAL AIRPORT LUNKEN FIELD, OH US	2017-12-27	13.00	5.80	3.82
CINCINNATI MUNICIPAL AIRPORT LUNKEN FIELD, OH US	2017-12-28	13.60	5.80	4.53
CINCINNATI MUNICIPAL AIRPORT LUNKEN FIELD, OH US	2017-01-06	13.60	5.50	4.87
CINCINNATI MUNICIPAL AIRPORT LUNKEN FIELD, OH US	2017-01-08	15.90	5.20	7.93
CINCINNATI MUNICIPAL AIRPORT LUNKEN FIELD, OH US	2017-12-25	25.80	13.50	14.29
CINCINNATI MUNICIPAL AIRPORT LUNKEN FIELD, OH US	2017-12-30	21.60	5.30	14.54
CINCINNATI MUNICIPAL AIRPORT LUNKEN FIELD, OH US	2017-01-05	22.20	5.80	14.75
CINCINNATI MUNICIPAL AIRPORT LUNKEN FIELD, OH US	2017-12-26	23.30	6.20	15.69

9.

| CINCINNATI MUNICIPAL AIRPORT LUNKEN FIELD, OH US | 2017-12-26 | 23.30 | 6.20 | 15.69 |

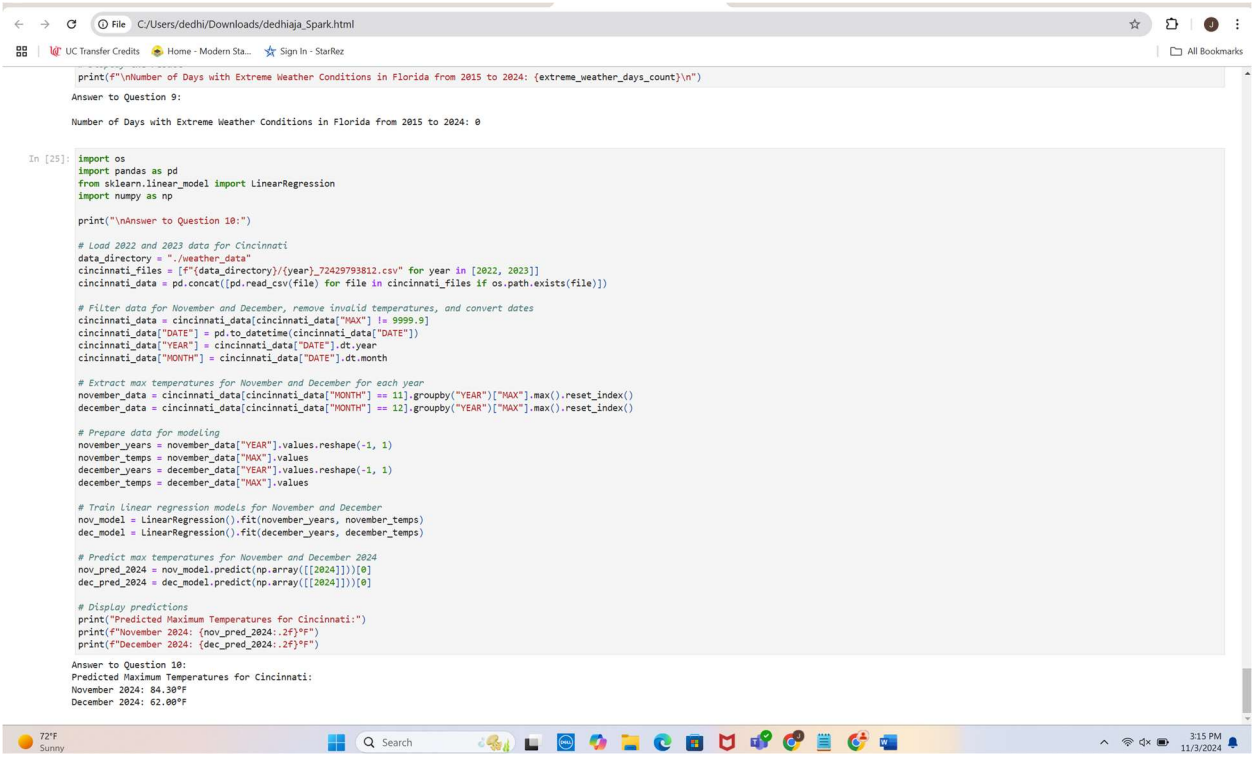
In [23]:
```
import os
import pandas as pd

print("\nAnswer to Question 9:")

# Directory and years for Florida files
data_directory = "./weather_data"
years = [y for y in range(2015, 2025) if y != 2016]  # Exclude 2016 if data is unavailable

# Initialize a counter for extreme weather days
extreme_weather_days_count = 0

# Load and process data for each year in the specified range
for year in years:
    florida_file = f"{data_directory}/{year}_99495199999.csv"

    if os.path.exists(florida_file):
        df = pd.read_csv(florida_file)

        # Ensure each FRSHTT value is a six-character string
        df['FRSHTT'] = df['FRSHTT'].astype(str).str.zfill(6)

        # Count days with any extreme weather indicator
        extreme_weather_days = df[df['FRSHTT'].apply(
            lambda x: any(x[i] == "1" for i in range(6))
        )]

        # Update the total count of extreme weather days
        extreme_weather_days_count += len(extreme_weather_days)

# Display the result
print(f"\nNumber of Days with Extreme Weather Conditions in Florida from 2015 to 2024: {extreme_weather_days_count}\n")
```

Answer to Question 9:

Number of Days with Extreme Weather Conditions in Florida from 2015 to 2024: 0

In [25]:
```
import os
import pandas as pd
from sklearn.linear_model import LinearRegression
import numpy as np

print("\nAnswer to Question 10:")

# Load 2022 and 2023 data for Cincinnati
data_directory = "./weather_data"
cincinnati_files = [f"{data_directory}/{year}_72429793812.csv" for year in [2022, 2023]]
cincinnati_data = pd.concat([pd.read_csv(file) for file in cincinnati_files if os.path.exists(file)])
```

10.



```
print(f"\nNumber of Days with Extreme Weather Conditions in Florida from 2015 to 2024: {extreme_weather_days_count}\n")
```

```
Answer to Question 9:

Number of Days with Extreme Weather Conditions in Florida from 2015 to 2024: 0
```

In [25]:

```
import os
import pandas as pd
from sklearn.linear_model import LinearRegression
import numpy as np

print("\nAnswer to Question 10:")

# Load 2022 and 2023 data for Cincinnati
data_directory = "./weather_data"
cincinnati_files = [f"{data_directory}/{year}_72429793812.csv" for year in [2022, 2023]]
cincinnati_data = pd.concat([pd.read_csv(file) for file in cincinnati_files if os.path.exists(file)])

# Filter data for November and December, remove invalid temperatures, and convert dates
cincinnati_data = cincinnati_data[cincinnati_data["MAX"] != 9999.9]
cincinnati_data["DATE"] = pd.to_datetime(cincinnati_data["DATE"])
cincinnati_data["YEAR"] = cincinnati_data["DATE"].dt.year
cincinnati_data["MONTH"] = cincinnati_data["DATE"].dt.month

# Extract max temperatures for November and December for each year
november_data = cincinnati_data[cincinnati_data["MONTH"] == 11].groupby("YEAR")["MAX"].max().reset_index()
december_data = cincinnati_data[cincinnati_data["MONTH"] == 12].groupby("YEAR")["MAX"].max().reset_index()

# Prepare data for modeling
november_years = november_data["YEAR"].values.reshape(-1, 1)
november_temps = november_data["MAX"].values
december_years = december_data["YEAR"].values.reshape(-1, 1)
december_temps = december_data["MAX"].values

# Train linear regression models for November and December
nov_model = LinearRegression().fit(november_years, november_temps)
dec_model = LinearRegression().fit(december_years, december_temps)

# Predict max temperatures for November and December 2024
nov_pred_2024 = nov_model.predict(np.array([[2024]]))[0]
dec_pred_2024 = dec_model.predict(np.array([[2024]]))[0]

# Display predictions
print("Predicted Maximum Temperatures for Cincinnati:")
print(f"November 2024: {nov_pred_2024:.2f}°F")
print(f"December 2024: {dec_pred_2024:.2f}°F")
```

```
Answer to Question 10:
Predicted Maximum Temperatures for Cincinnati:
November 2024: 84.30°F
December 2024: 62.00°F
```

Using a linear regression model trained on 2022 and 2023 maximum temperature data, the predicted maximum temperatures for Cincinnati in 2024 are 84.30°F in November and 62.00°F in December. This simple model uses the temperature trend over the previous two years to forecast values for the upcoming season, offering a basic yet effective approach given the limited historical data.

However, this model has limitations due to its simplicity and lack of seasonal adjustments. For improved accuracy, a more comprehensive model could incorporate additional years of historical data, seasonal patterns, or other weather variables like humidity and wind speed. More advanced time-series models, such as SARIMA or Exponential Smoothing, could better capture seasonal effects, yielding more reliable forecasts.