Aayush Keshari

CS 5165

PROJECT 4: Big Data with PySpark using Anaconda & Jupyter notebook

1.

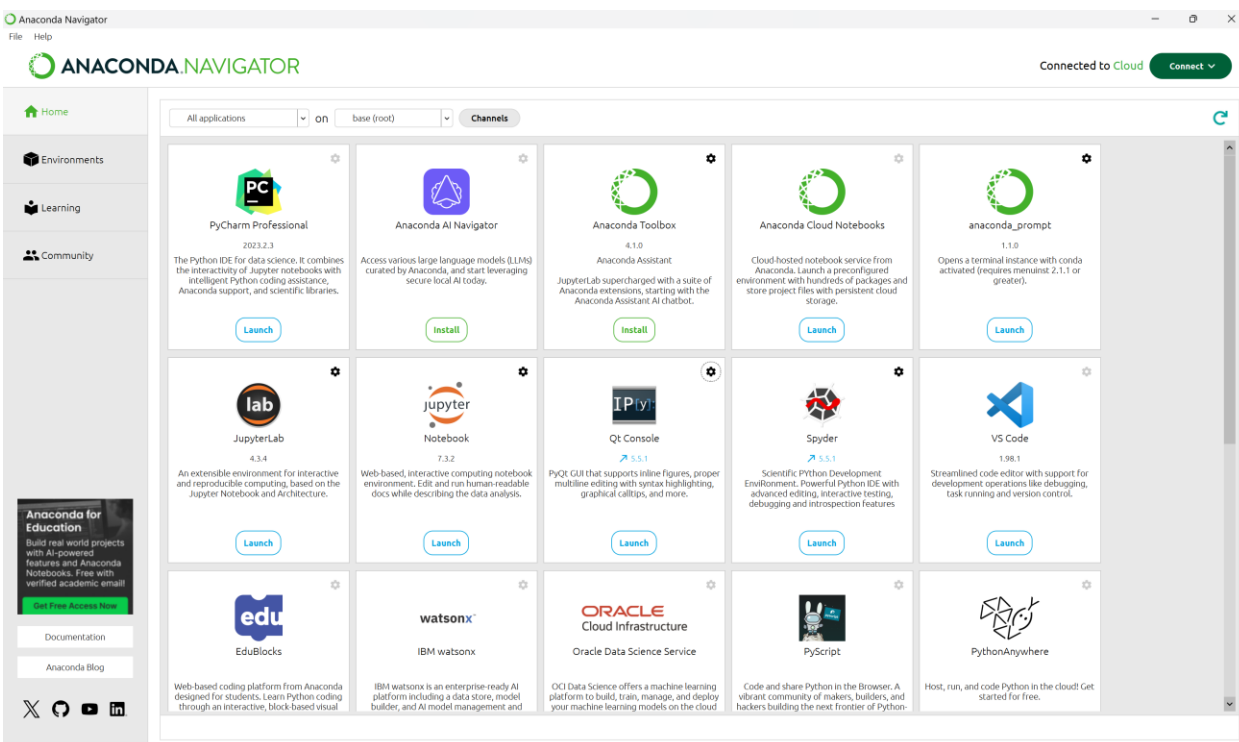
[2]:
```
# Name: Aayush Keshari
# Date: March 23, 2025
# CS 5165
# Project 4 - Big Data
```

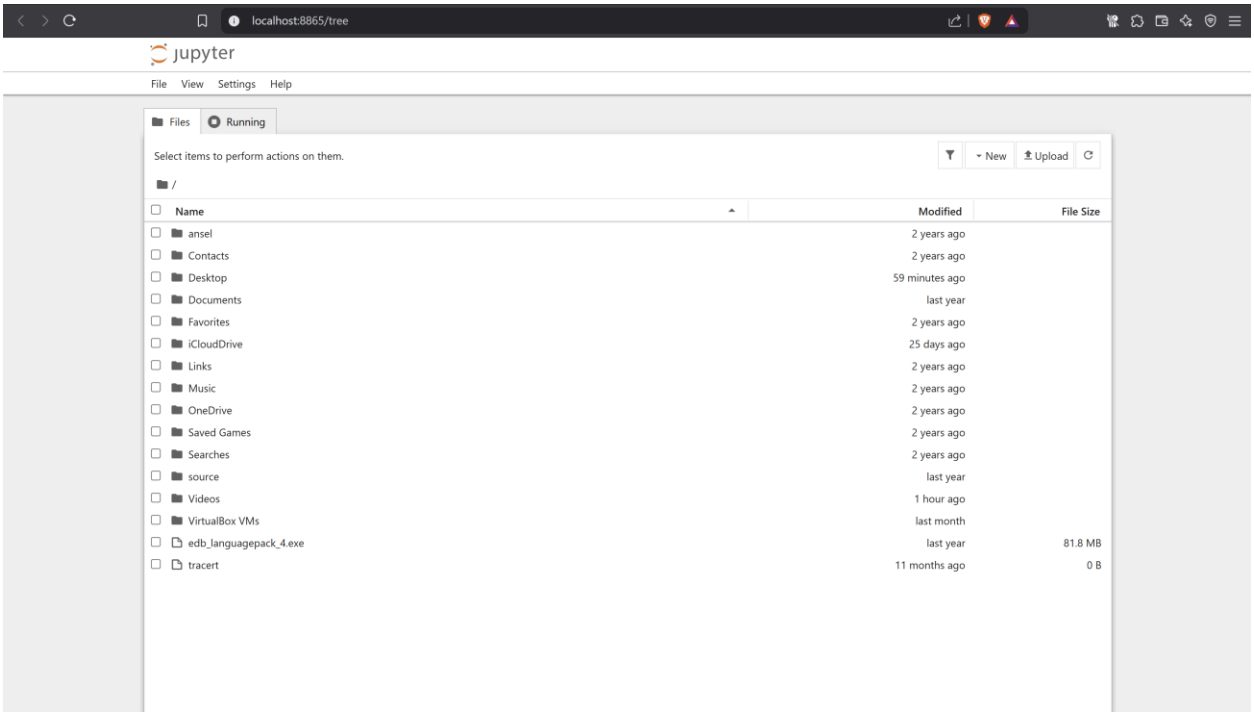
[3]:
```
!pip install pyspark
```

```
^C
```

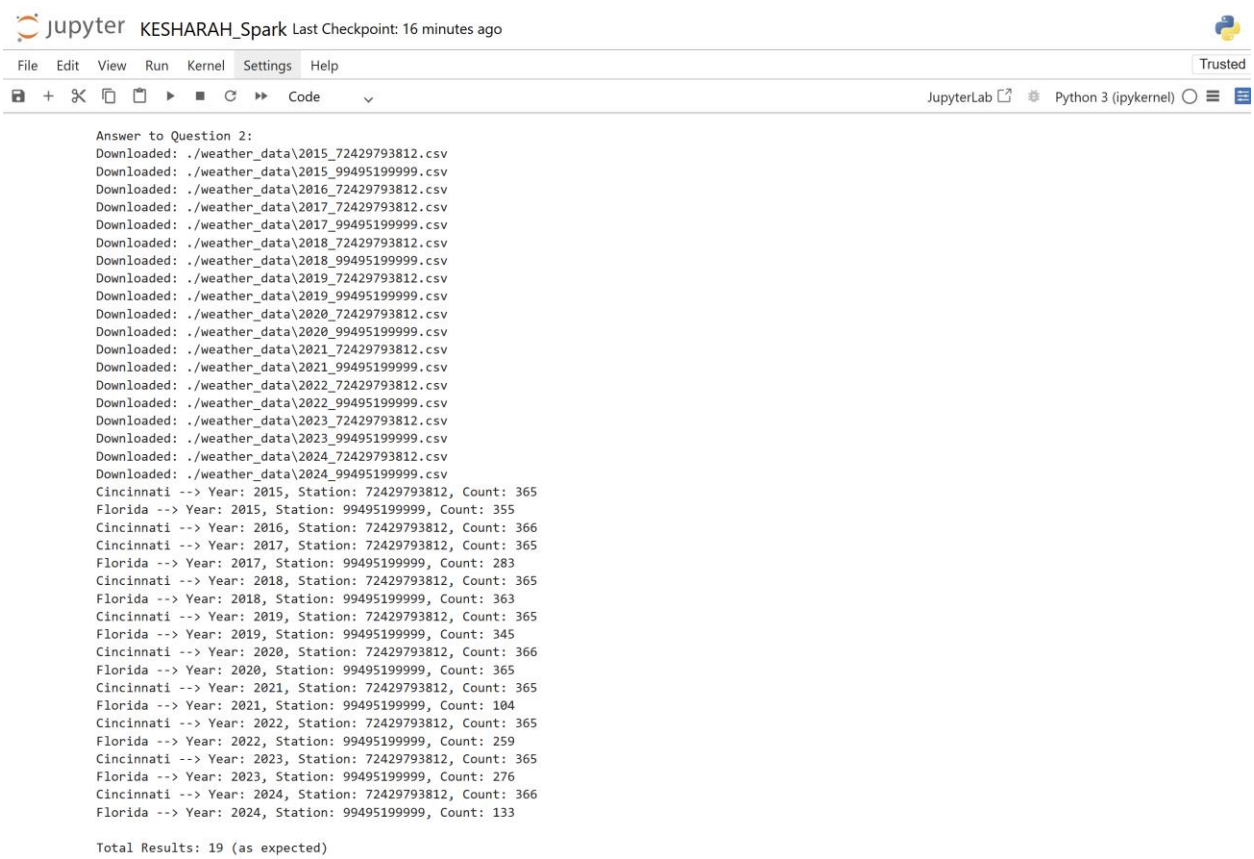
[4]:
```
!pip install requests beautifulsoup4
```

```
^C
Requirement already satisfied: requests in c:\programdata\anaconda3\lib\site-packages (2.32.3)
Requirement already satisfied: beautifulsoup4 in c:\programdata\anaconda3\lib\site-packages (4.12.3)
Requirement already satisfied: charset-normalizer<4,>=2 in c:\programdata\anaconda3\lib\site-packages (from requests) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in c:\programdata\anaconda3\lib\site-packages (from requests) (3.7)
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\programdata\anaconda3\lib\site-packages (from requests) (2.2.3)
Requirement already satisfied: certifi>=2017.4.17 in c:\programdata\anaconda3\lib\site-packages (from requests) (2025.1.31)
Requirement already satisfied: soupsieve>1.2 in c:\programdata\anaconda3\lib\site-packages (from beautifulsoup4) (2.5)
```




localhost:8865/tree
jupyter
File View Settings Help
Files
Running
Select items to perform actions on them.
New
Upload
/
Name
Modified
File Size
ansel 2 years ago
Contacts 2 years ago
Desktop 59 minutes ago
Documents last year
Favorites 2 years ago
iCloudDrive 25 days ago
Links 2 years ago
Music 2 years ago
OneDrive 2 years ago
Saved Games 2 years ago
Searches 2 years ago
source last year
Videos 1 hour ago
VirtualBox VMs last month
edb_languagepack_4.exe last year 81.8 MB
tracert 11 months ago 0 B

2.


jupyter KESHARAH_Spark Last Checkpoint: 16 minutes ago
File Edit View Run Kernel Settings Help
Trusted
Code
JupyterLab
Python 3 (ipykernel)
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: 366
Florida --> Year: 2024, Station: 99495199999, Count: 133
Total Results: 19 (as expected)

jupyter KESHARAH_Spark Last Checkpoint: 18 minutes ago

File Edit View Run Kernel Settings Help Trusted

Code JupyterLab Python 3 (ipykernel)

[6]:
```python
import os
import pandas as pd

# Directory containing the CSV files
data_directory = "./weather_data"

# Years for which data is available
years = range(2015, 2025)

# List CSV files for Cincinnati and Florida
cincinnati_files = [f"{data_directory}/{year}_72429793812.csv" for year in years]
florida_files = [f"{data_directory}/{year}_99495199999.csv" for year in years if year != 2016]  # Exclude 2016 for Florida

# 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): 3653
Florida Data Count (Total Number of Rows)   : 2483
```

3.

jupyter KESHARAH_Spark Last Checkpoint: 23 minutes ago

File Edit View Run Kernel Settings Help Trusted

Code JupyterLab Python 3 (ipykernel)

```
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

jupyter KESHARAH_Spark Last Checkpoint: 24 minutes ago

File Edit View Run Kernel Settings Help Trusted

Code JupyterLab Python 3 (ipykernel)

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.

jupyter KESHARAH_Spark Last Checkpoint: 26 minutes ago

File Edit View Run Kernel Settings Help Trusted

Code JupyterLab Python 3 (ipykernel)

```
            # Find the coldest day in March for this file, if available
            if not march_df.empty:
                coldest_day = march_df.loc[march_df["MIN"].idxmin()]
                march_min_temps.append({
                    "YEAR": year,
                    "STATION": str(coldest_day["STATION"]),  # Ensure Station ID is stored as a string
                    "NAME": coldest_day["NAME"],
                    "DATE": coldest_day["DATE"],
                    "MIN": coldest_day["MIN"]
                })

# Convert the list to a DataFrame and find the overall coldest day in March across all years
all_march_min_df = pd.DataFrame(march_min_temps)
coldest_overall_day = all_march_min_df.loc[all_march_min_df["MIN"].idxmin()]

# Prepare data for display
results = [
    {
        "Year": int(coldest_overall_day["YEAR"]),
        "Station ID": coldest_overall_day["STATION"],
        "Station Name": coldest_overall_day["NAME"],
        "Date": coldest_overall_day["DATE"].strftime("%Y-%m-%d"),
        "Min Temp (°F)": round(coldest_overall_day["MIN"], 1)
    }
]

# Display the result in a well-formatted table
print("\nColdest Day Overall in March (2015-2024) across Cincinnati and Florida:\n")
print(tabulate(results, headers="keys", tablefmt="fancy_grid"))
```

```
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.

jupyter KESHARAH_Spark Last Checkpoint: 28 minutes ago

File Edit View Run Kernel Settings Help Trusted

Code JupyterLab Python 3 (ipykernel)

```python
                "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	4.5
2015	99495199999	SEBASTIAN INLET STATE PARK, FL US	0

6.

jupyter KESHARAH_Spark Last Checkpoint: 28 minutes ago

File Edit View Run Kernel Settings Help Trusted

Code JupyterLab Python 3 (ipykernel)

```python
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: 39.07%
Florida: 100.00%
```

7.

jupyter KESHARAH_Spark Last Checkpoint: 31 minutes ago

File Edit View Run Kernel Settings Help Trusted

Code JupyterLab Python 3 (ipykernel)

```
    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.

jupyter KESHARAH_Spark Last Checkpoint: 31 minutes ago

File Edit View Run Kernel Settings Help Trusted

Code JupyterLab Python 3 (ipykernel)

```
        35.74 + (0.6215 * filtered_df["TEMP"]) - (35.75 * (filtered_df["WDSP"] ** 0.16)) +
        (0.4275 * filtered_df["TEMP"] * (filtered_df["WDSP"] ** 0.16))
    )

    # Get the top 10 days with the lowest Wind Chill
    top_10_lowest_wc = filtered_df.nsmallest(10, "Wind_Chill")[["NAME", "DATE", "TEMP", "WDSP", "Wind_Chill"]]

    # Display results in a well-formatted table
    print("\nTop 10 Days with the Lowest Wind Chill for Cincinnati in 2017:\n")
    print(tabulate(top_10_lowest_wc, headers="keys", tablefmt="fancy_grid", floatfmt=".2f", showindex=False))
else:
    print("Cincinnati 2017 data file not found.")
```

```
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.

jupyter KESHARAH_Spark Last Checkpoint: 33 minutes ago

File Edit View Run Kernel Settings Help Trusted

Code JupyterLab Python 3 (ipykernel)

[13]:
```python
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
```

10.

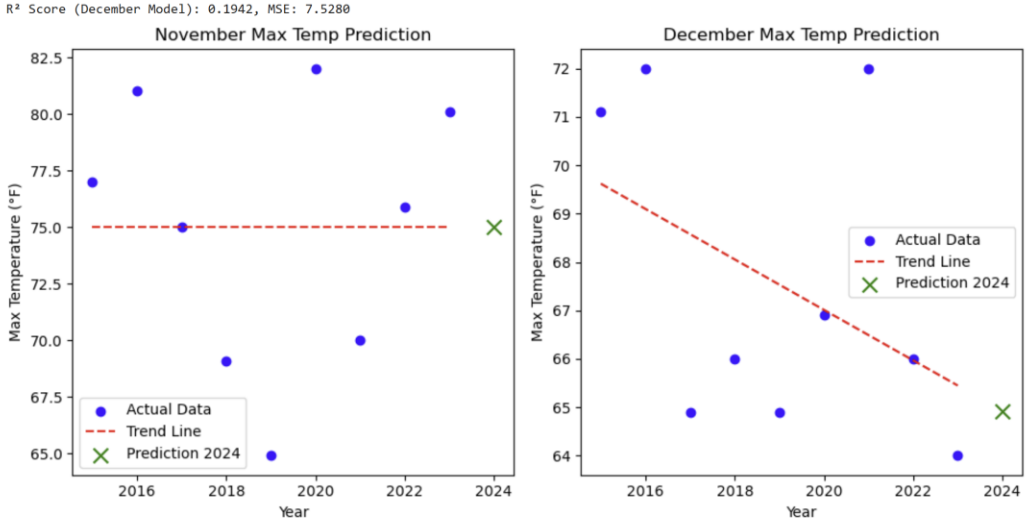
jupyter KESHARAH_Spark Last Checkpoint: 34 minutes ago

File Edit View Run Kernel Settings Help Trusted

Code JupyterLab Python 3 (ipykernel)

```
Answer to Question 10:
Predicted Maximum Temperatures for Cincinnati:
November 2024: 75.00°F
December 2024: 64.92°F

Model Performance:
R² Score (November Model): 0.0000, MSE: 30.8489
R² Score (December Model): 0.1942, MSE: 7.5280
```



The model's performance, particularly for November, is weak, as indicated by the R^2 score of 0.0000, meaning it fails to capture any meaningful trend in the data. This suggests that November's temperatures may be highly variable or influenced by non-linear patterns that a simple linear regression cannot model. The December model performs slightly better ($R^2 = 0.1942$), showing a weak downward trend, but still lacks strong predictive power. The high Mean Squared Errors (MSEs) further indicate that the models struggle with accuracy. To improve predictions, incorporating more historical data could help capture long-term trends. Additionally, non-linear models, such as polynomial regression or time series models like ARIMA or SARIMA, might better account for seasonal patterns and fluctuations. Using external factors, such as climate indices or weather anomalies, could also enhance model reliability.

GitHub Repository: https://github.com/aayushkeshari/Project-4-Big-Data