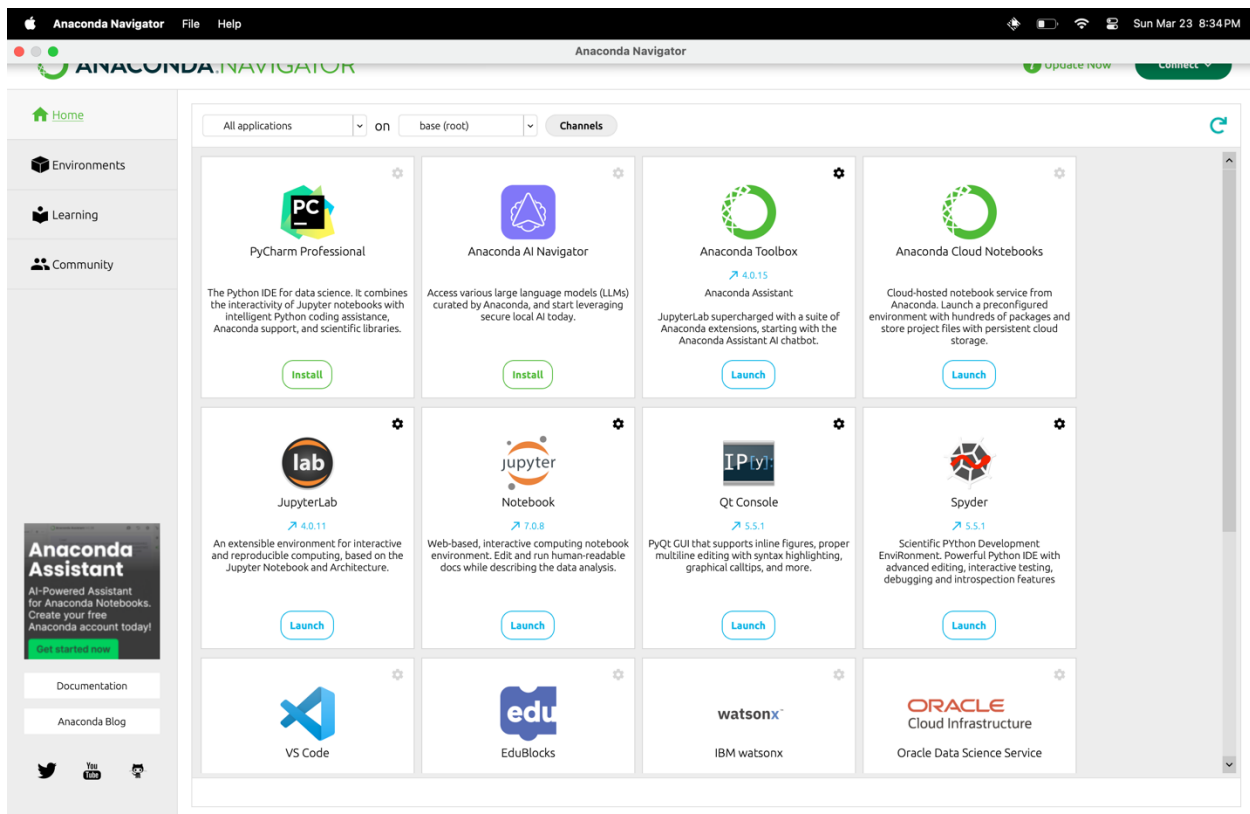


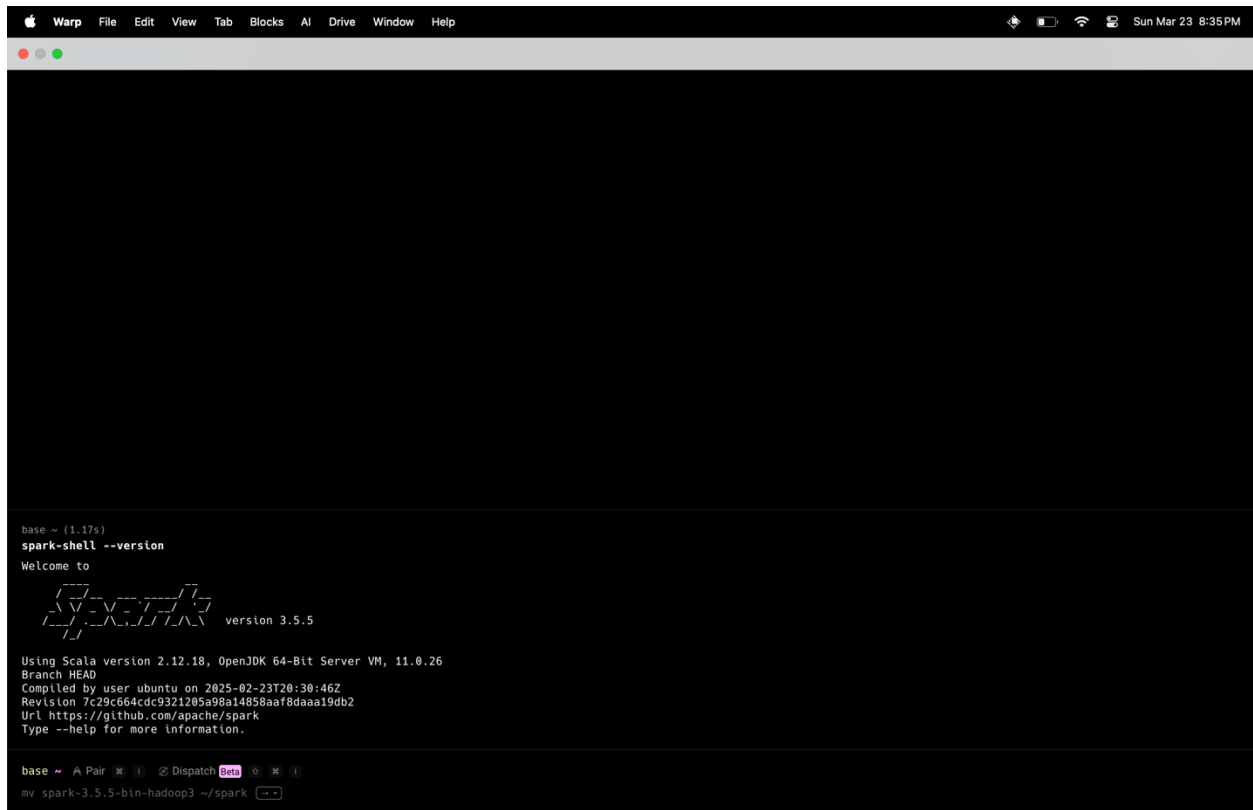
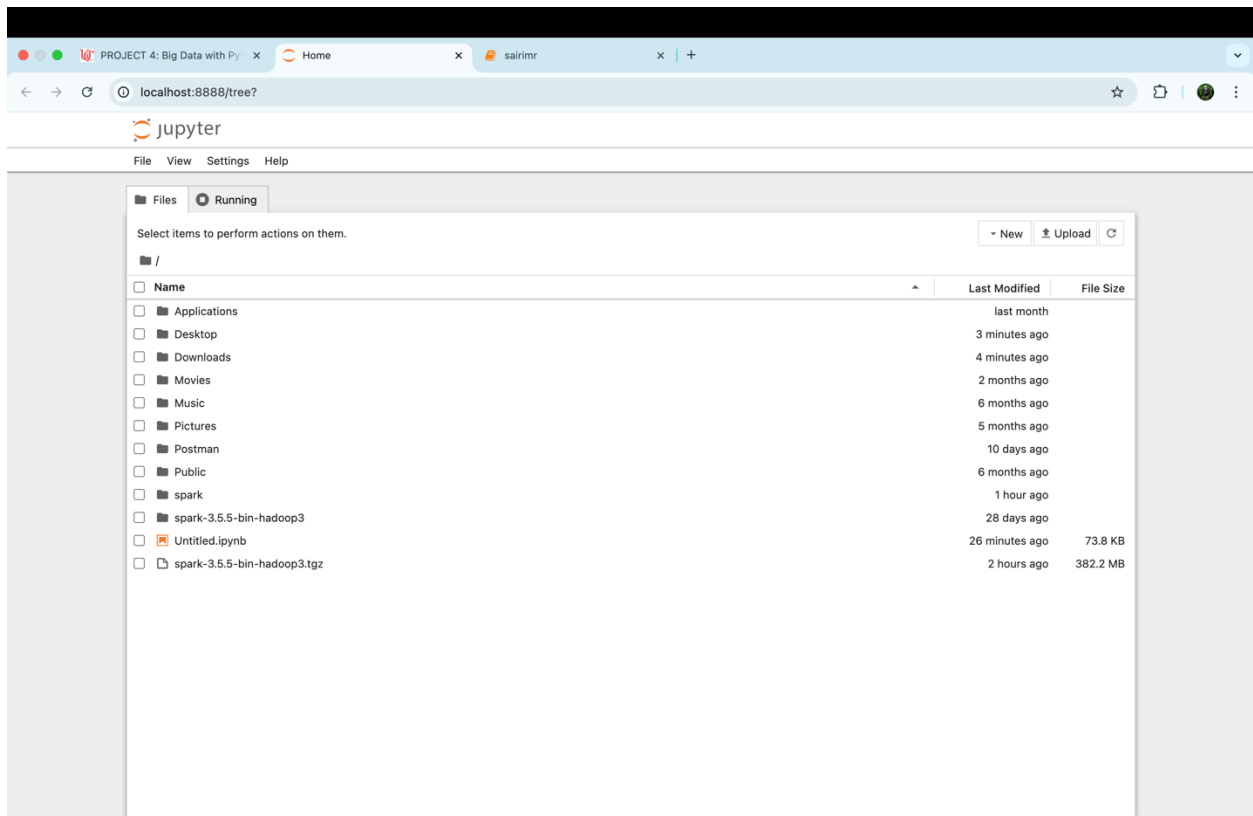
PROJECT 4: Big Data with PySpark using Anaconda & Jupyter notebook

Muralidhar Sairi

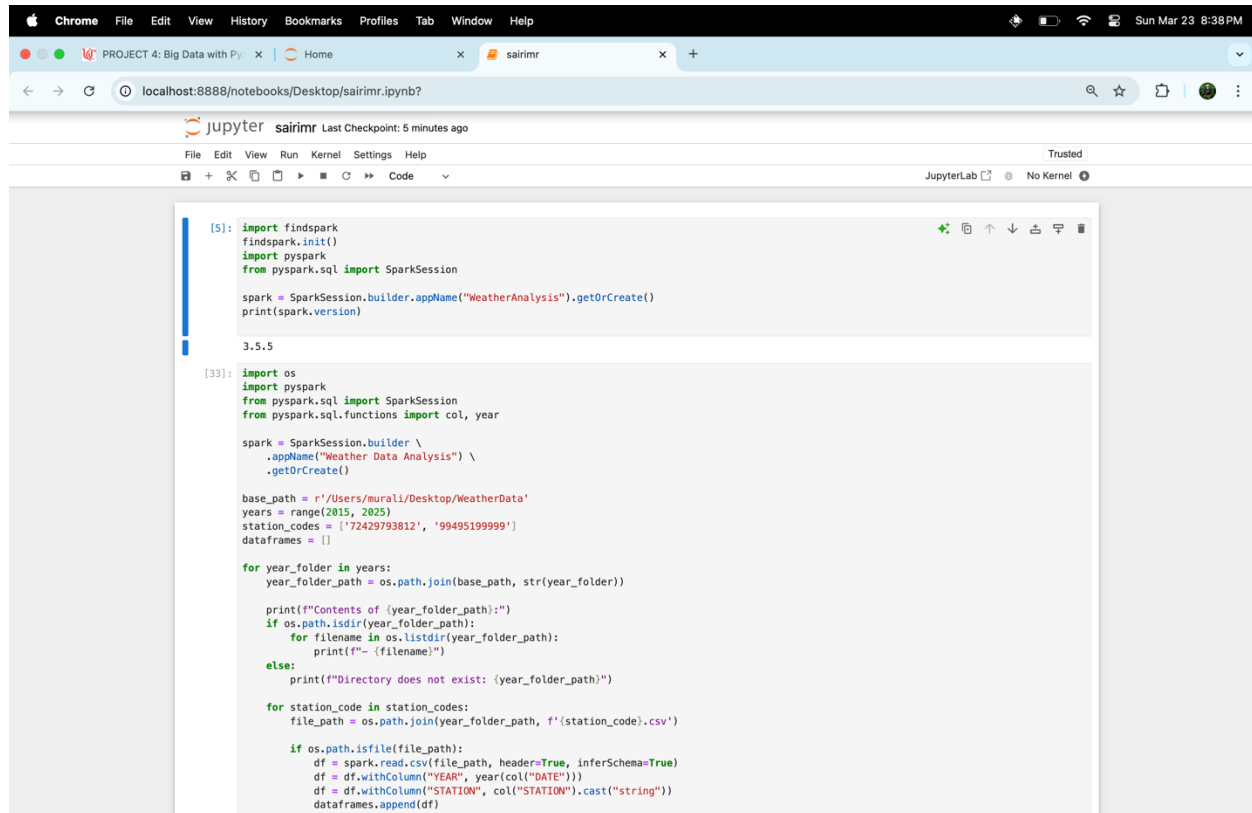
M16503614

Download Anaconda Python, install PySpark, and launch Jupyter Notebook:





Load the CSV files and display the count of each dataset:



The screenshot shows a JupyterLab interface in a Chrome browser. The address bar shows the URL `localhost:8888/notebooks/Desktop/sairimr.ipynb?`. The JupyterLab header shows the name `sairimr` and a status of `Trusted`. The code editor displays two code cells. The first cell, labeled `[5]:`, contains the following Python code:

```
[5]: import findspark
findspark.init()
import pyspark
from pyspark.sql import SparkSession

spark = SparkSession.builder.appName("WeatherAnalysis").getOrCreate()
print(spark.version)

3.5.5
```

The second cell, labeled `[33]:`, contains the following Python code:

```
[33]: import os
import pyspark
from pyspark.sql import SparkSession
from pyspark.sql.functions import col, year

spark = SparkSession.builder \
    .appName("Weather Data Analysis") \
    .getOrCreate()

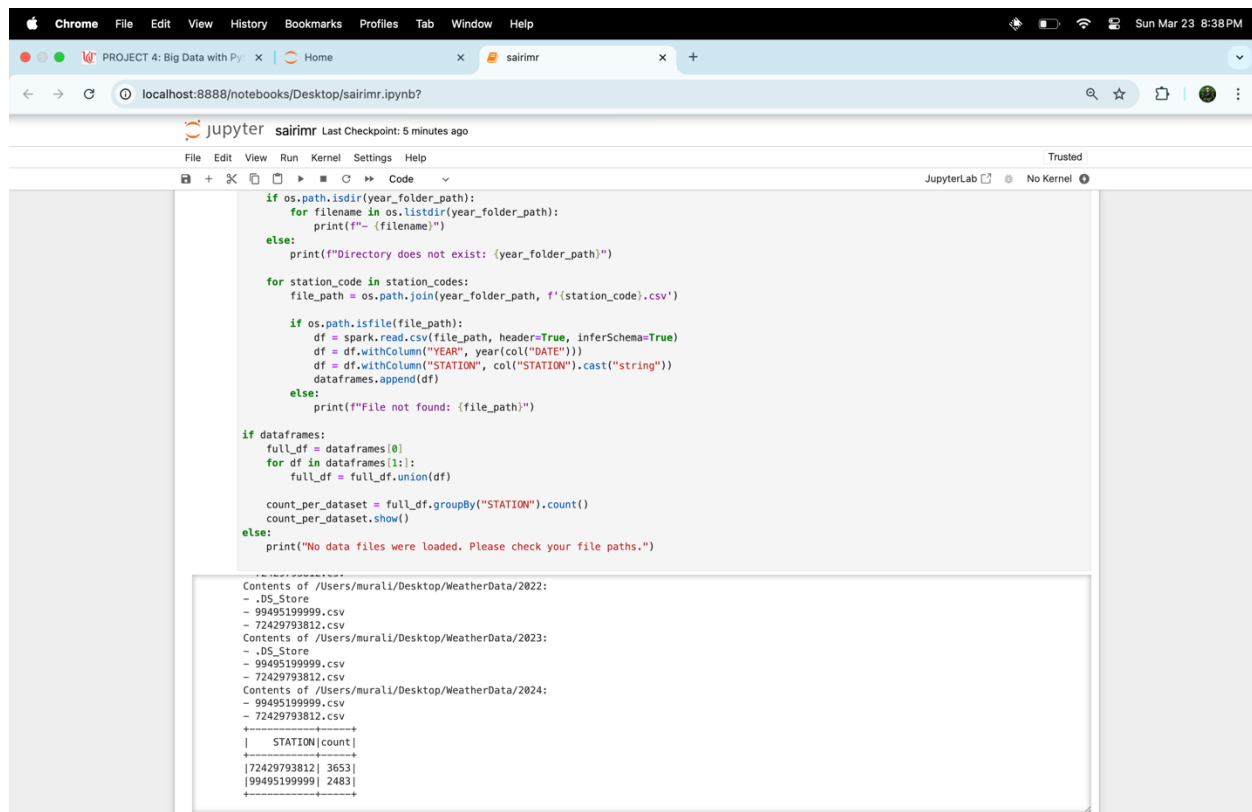
base_path = r'/Users/murali/Desktop/WeatherData'
years = range(2015, 2025)
station_codes = ['72429793812', '99495199999']
dataframes = []

for year_folder in years:
    year_folder_path = os.path.join(base_path, str(year_folder))

    print(f"Contents of {year_folder_path}:")
    if os.path.isdir(year_folder_path):
        for filename in os.listdir(year_folder_path):
            print(f"~ {filename}")
        else:
            print(f"Directory does not exist: {year_folder_path}")

    for station_code in station_codes:
        file_path = os.path.join(year_folder_path, f'{station_code}.csv')

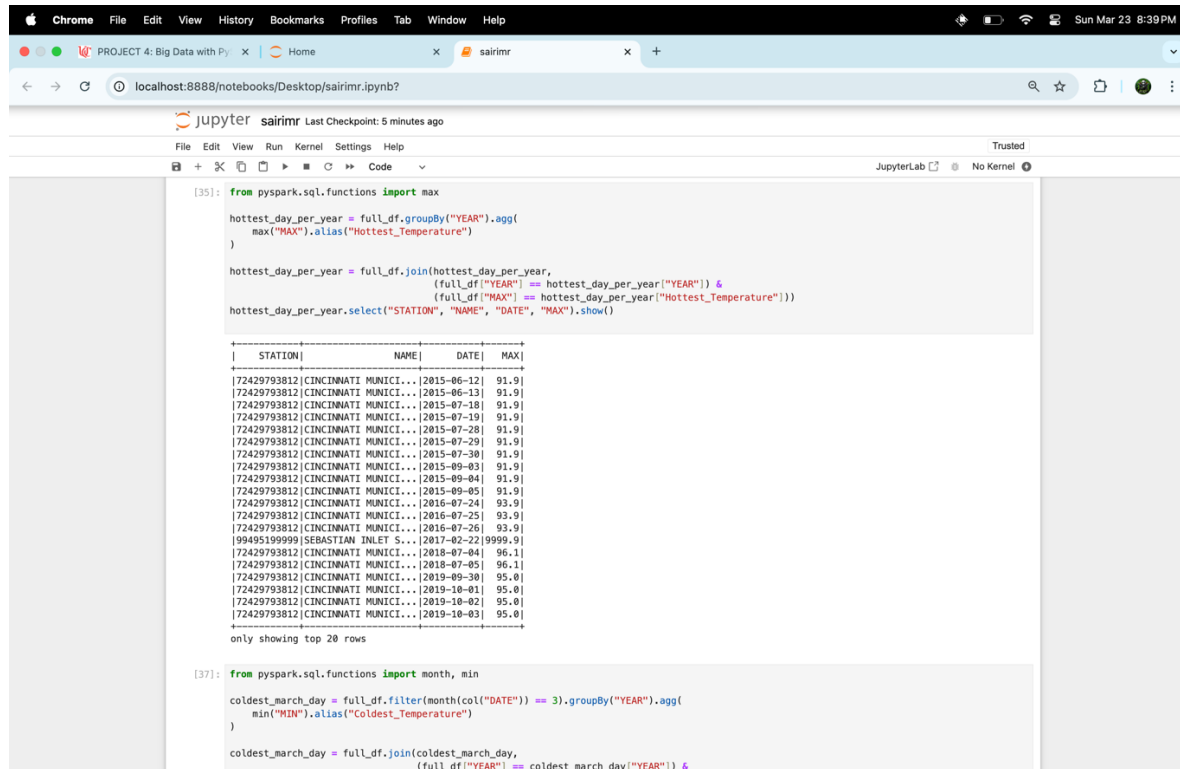
        if os.path.isfile(file_path):
            df = spark.read.csv(file_path, header=True, inferSchema=True)
            df = df.withColumn("YEAR", year(col("DATE")))
            df = df.withColumn("STATION", col("STATION").cast("string"))
            dataframes.append(df)
        else:
            print(f"File not found: {file_path}")
```



The screenshot shows the same JupyterLab interface, but the second code cell is now executed, and its output is displayed in the output area below the code editor. The output shows the contents of the directories and the resulting dataframes.

```
Contents of /Users/murali/Desktop/WeatherData/2022:
- .DS_Store
- 99495199999.csv
- 72429793812.csv
Contents of /Users/murali/Desktop/WeatherData/2023:
- .DS_Store
- 99495199999.csv
- 72429793812.csv
Contents of /Users/murali/Desktop/WeatherData/2024:
- 99495199999.csv
- 72429793812.csv
+-----+-----+
| STATION|count|
+-----+-----+
|72429793812| 3653|
|99495199999| 2483|
+-----+-----+
```

Find the hottest day (column MAX) for each year:



The screenshot shows a JupyterLab notebook interface with a Chrome browser window at the top. The notebook is titled 'sairimr' and shows a PySpark SQL query in cell [35]. The query finds the hottest day for each year by grouping by year and selecting the maximum temperature. The result is displayed as a table with columns STATION, NAME, DATE, and MAX. The table shows data for various stations from 2015 to 2019, with the highest temperature being 96.1 degrees Fahrenheit on 2018-07-04 at station 72429793812. The notebook also shows a second query in cell [37] that finds the coldest day for the month of March across all years from 2015 to 2024.

```
[35]: from pyspark.sql.functions import max

hottest_day_per_year = full_df.groupBy("YEAR").agg(
    max("MAX").alias("Hottest_Temperature")
)

hottest_day_per_year = full_df.join(hottest_day_per_year,
    (full_df["YEAR"] == hottest_day_per_year["YEAR"]) &
    (full_df["MAX"] == hottest_day_per_year["Hottest_Temperature"]))
hottest_day_per_year.select("STATION", "NAME", "DATE", "MAX").show()

+-----+-----+-----+-----+
| STATION | NAME | DATE | MAX |
+-----+-----+-----+-----+
| 72429793812 | CININNATI MUNICI... | 2015-06-12 | 91.9 |
| 72429793812 | CININNATI MUNICI... | 2015-06-13 | 91.9 |
| 72429793812 | CININNATI MUNICI... | 2015-07-18 | 91.9 |
| 72429793812 | CININNATI MUNICI... | 2015-07-19 | 91.9 |
| 72429793812 | CININNATI MUNICI... | 2015-07-28 | 91.9 |
| 72429793812 | CININNATI MUNICI... | 2015-07-29 | 91.9 |
| 72429793812 | CININNATI MUNICI... | 2015-07-30 | 91.9 |
| 72429793812 | CININNATI MUNICI... | 2015-09-03 | 91.9 |
| 72429793812 | CININNATI MUNICI... | 2015-09-04 | 91.9 |
| 72429793812 | CININNATI MUNICI... | 2015-09-05 | 91.9 |
| 72429793812 | CININNATI MUNICI... | 2016-07-24 | 93.9 |
| 72429793812 | CININNATI MUNICI... | 2016-07-25 | 93.9 |
| 72429793812 | CININNATI MUNICI... | 2016-07-26 | 93.9 |
| 99495199999 | SEBASTIAN INLET S... | 2017-02-22 | 9999.9 |
| 72429793812 | CININNATI MUNICI... | 2018-07-04 | 96.1 |
| 72429793812 | CININNATI MUNICI... | 2018-07-05 | 96.1 |
| 72429793812 | CININNATI MUNICI... | 2019-09-30 | 95.0 |
| 72429793812 | CININNATI MUNICI... | 2019-10-01 | 95.0 |
| 72429793812 | CININNATI MUNICI... | 2019-10-02 | 95.0 |
| 72429793812 | CININNATI MUNICI... | 2019-10-03 | 95.0 |
+-----+-----+-----+-----+
only showing top 20 rows

[37]: from pyspark.sql.functions import month, min

coldest_march_day = full_df.filter(month(col("DATE")) == 3).groupBy("YEAR").agg(
    min("MIN").alias("Coldest_Temperature")
)

coldest_march_day = full_df.join(coldest_march_day,
    (full_df["YEAR"] == coldest_march_day["YEAR"]) &
```

Find the coldest day (column MIN) for the month of March across all years (2015-2024) :

Chrome File Edit View History Bookmarks Profiles Tab Window Help

PROJECT 4: Big Data with Py x Home x sairmr x +

localhost:8888/notebooks/Desktop/sairmr.ipynb?

jupyter sairmr Last Checkpoint: 6 minutes ago

File Edit View Run Kernel Settings Help Trusted

only showing top 20 rows

```
[37]: from pyspark.sql.functions import month, min

coldest_march_day = full_df.filter(month(col("DATE")) == 3).groupBy("YEAR").agg(
    min("MIN").alias("Coldest_Temperature")
)

coldest_march_day = full_df.join(coldest_march_day,
                                (full_df["YEAR"] == coldest_march_day["YEAR"]) &
                                (full_df["MIN"] == coldest_march_day["Coldest_Temperature"]))
coldest_march_day.select("STATION", "NAME", "DATE", "MIN").show(1)
```

STATION	NAME	DATE	MIN
72429793812	CINCINNATI MUNICI...	2015-03-06	3.2

only showing top 1 row

Find the year with the most precipitation for Cincinnati and Florida:

Chrome File Edit View History Bookmarks Profiles Tab Window Help

PROJECT 4: Big Data with Py x Home x sairmr x +

localhost:8888/notebooks/Desktop/sairmr.ipynb?

jupyter sairmr Last Checkpoint: 18 minutes ago

File Edit View Run Kernel Settings Help Trusted

only showing top 1 row

```
[24]: precipitation_data = full_df.filter(full_df["STATION"].isin(["72429793812", "99495199999"]))

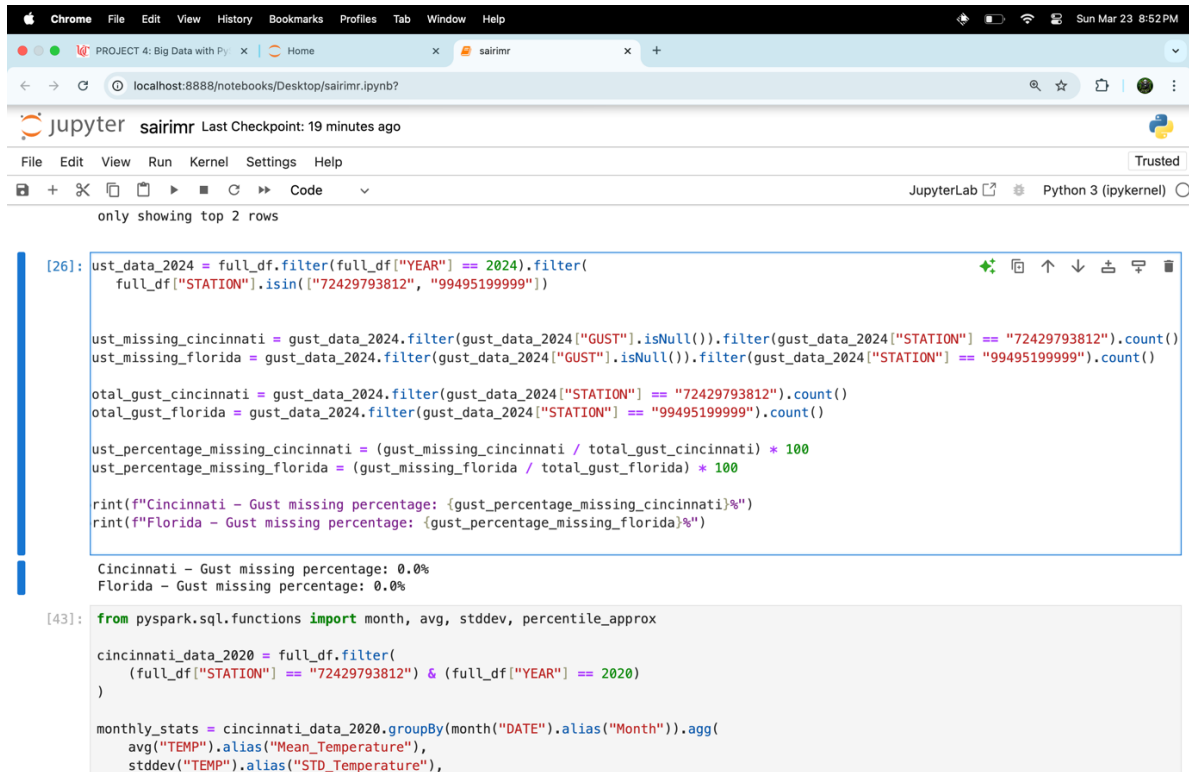
precipitation_per_year = precipitation_data.groupBy("STATION", "YEAR").agg(
    {"PRCP": "mean"}
).withColumnRenamed("avg(PRCP)", "Mean_of_PRCP")

precipitation_per_year.show(2)
```

STATION	YEAR	Mean_of_PRCP
72429793812	2015	0.13178082191780816
99495199999	2015	0.0

only showing top 2 rows

Count the percentage of missing values for wind gust (column GUST) for Cincinnati and Florida in the year 2024:



The screenshot shows a JupyterLab interface with a browser window at the top displaying the URL `localhost:8888/notebooks/Desktop/sairimr.ipynb?`. The JupyterLab header includes the Jupyter logo, the name 'sairimr', and 'Last Checkpoint: 19 minutes ago'. Below the header is a menu bar with 'File', 'Edit', 'View', 'Run', 'Kernel', 'Settings', and 'Help'. A toolbar contains icons for file operations and a 'Code' dropdown menu. The main area shows a Jupyter Notebook with two code cells. The first cell, labeled '[26]:', contains Python code that filters data for the year 2024, calculates the number of missing values for 'GUST' in Cincinnati and Florida, and then calculates the percentage of missing values. The output of this cell shows 'Cincinnati - Gust missing percentage: 0.0%' and 'Florida - Gust missing percentage: 0.0%'. The second cell, labeled '[43]:', contains Python code that imports functions from 'pyspark.sql.functions' and filters data for the year 2020 to calculate monthly statistics for temperature.

```
[26]: ust_data_2024 = full_df.filter(full_df["YEAR"] == 2024).filter(
      full_df["STATION"].isin(["72429793812", "99495199999"])

      ust_missing_cincinnati = gust_data_2024.filter(gust_data_2024["GUST"].isNull()).filter(gust_data_2024["STATION"] == "72429793812").count()
      ust_missing_florida = gust_data_2024.filter(gust_data_2024["GUST"].isNull()).filter(gust_data_2024["STATION"] == "99495199999").count()

      total_gust_cincinnati = gust_data_2024.filter(gust_data_2024["STATION"] == "72429793812").count()
      total_gust_florida = gust_data_2024.filter(gust_data_2024["STATION"] == "99495199999").count()

      ust_percentage_missing_cincinnati = (gust_missing_cincinnati / total_gust_cincinnati) * 100
      ust_percentage_missing_florida = (gust_missing_florida / total_gust_florida) * 100

      print(f"Cincinnati - Gust missing percentage: {gust_percentage_missing_cincinnati}%")
      print(f"Florida - Gust missing percentage: {gust_percentage_missing_florida}%")

Cincinnati - Gust missing percentage: 0.0%
Florida - Gust missing percentage: 0.0%

[43]: from pyspark.sql.functions import month, avg, stddev, percentile_approx

      cincinnati_data_2020 = full_df.filter(
          (full_df["STATION"] == "72429793812") & (full_df["YEAR"] == 2020)
      )

      monthly_stats = cincinnati_data_2020.groupBy(month("DATE").alias("Month")).agg(
          avg("TEMP").alias("Mean_Temperature"),
          stddev("TEMP").alias("STD_Temperature"),
```

Find the mean, median, mode, and standard deviation of the temperature (column TEMP) for Cincinnati in each month for the year 2020:

```
[43]: from pyspark.sql.functions import month, avg, stddev, percentile_approx

cincinnati_data_2020 = full_df.filter(
    (full_df["STATION"] == "72429793812") & (full_df["YEAR"] == 2020)
)

monthly_stats = cincinnati_data_2020.groupBy(month("DATE").alias("Month")).agg(
    avg("TEMP").alias("Mean_Temperature"),
    stddev("TEMP").alias("STD_Temperature"),
    percentile_approx("TEMP", 0.5).alias("Median_Temperature")
)

monthly_stats.show(12)
```

Month	Mean_Temperature	STD_Temperature	Median_Temperature
12	35.99354838709677	6.642787340861814	35.2
1	37.94516129032259	8.345810873712928	37.7
6	72.54666666666667	4.899946047087439	73.7
3	49.0741935483871	8.779406500135623	47.8
5	60.89032258064518	9.314768017820217	63.7
9	66.1	7.118262089331474	65.8
4	51.779999999999994	7.313162436838541	51.0
8	73.34516129032258	3.487868375734898	73.7
7	77.6	2.33794781806609	77.9
10	55.193548387096776	6.72869157582517	54.0
11	48.003333333333345	6.825938527529321	47.7
2	36.5896551724138	7.90159770587055	36.0

Investigate how many days had extreme weather conditions for Florida (fog, rain, snow, etc.) using the **FRSHTT** column:

```
[47]: florida_data = full_df.filter(full_df["STATION"] == "99495199999")

extreme_weather_days = florida_data.filter(
    florida_data["FRSHTT"].isin(["1", "2", "3", "4", "5"])
)

extreme_weather_days_count = extreme_weather_days.count()
print(f"Number of days with extreme weather conditions: {extreme_weather_days_count}")

Number of days with extreme weather conditions: 0
```

```
[49]: from pyspark.ml.regression import LinearRegression
from pyspark.ml.feature import VectorAssembler
```

Find the top 10 days with the lowest Wind Chill for Cincinnati in 2017:

Chrome PROJECT 4: Big Data with Py: x Home x sairimr x +

localhost:8888/notebooks/Desktop/sairimr.ipynb?

Jupyter sairimr Last Checkpoint: 22 minutes ago

File Edit View Run Kernel Settings Help Trusted

JupyterLab Python 3 (ipykernel)

```
[45]: import pyspark.sql.functions as F

def calculate_wind_chill(temp, wind_speed):
    return 35.74 + 0.6215 * temp - 35.75 * (wind_speed**0.16) + 0.4275 * temp * (wind_speed**0.16)

wc_udf = F.udf(calculate_wind_chill, pyspark.sql.types.FloatType())

cincinnati_2017 = full_df.filter(
    (full_df["STATION"] == "72429793812") & (full_df["YEAR"] == 2017)
)

cincinnati_2017 = cincinnati_2017.filter(
    (cincinnati_2017["TEMP"] < 50) & (cincinnati_2017["WDSP"] > 3)
)

cincinnati_2017 = cincinnati_2017.withColumn("Wind_Chill", wc_udf(cincinnati_2017["TEMP"], cincinnati_2017["WDSP"]))
cincinnati_2017.orderBy("Wind_Chill").show(10)
```

25/03/23 20:30:20 WARN SparkStringUtils: Truncated the string representation of a plan since it was too large. This behavior can be adjusted by setting 'spark.sql.debug.maxToStringFields'.

STATION	DATE	LATITUDE	LONGITUDE	ELEVATION	NAME	TEMP	TEMP_ATTRIBUTES	DEWP	DEWP_ATTRIBUTES	SLP	SLP_ATTRIBUTES		
STP	STP_ATTRIBUTES	VISIB	VISIB_ATTRIBUTES	WDSP	WDSP_ATTRIBUTES	MXSPD	GUST	MAX	MAX_ATTRIBUTES	MIN	MIN_ATTRIBUTES	PRCP	PRCP_ATTRIBUTES
S	SNDP	FRSHTT	YEAR	Wind_Chill									

Chrome PROJECT 4: Big Data with Py: x Home x sairimr x +

localhost:8888/notebooks/Desktop/sairimr.ipynb?

Jupyter sairimr Last Checkpoint: 23 minutes ago

File Edit View Run Kernel Settings Help Trusted

JupyterLab Python 3 (ipykernel)

STATION	DATE	LATITUDE	LONGITUDE	ELEVATION	NAME	TEMP	TEMP_ATTRIBUTES	DEWP	DEWP_ATTRIBUTES	SLP	SLP_ATTRIBUTES	S		
TP	STP_ATTRIBUTES	VISIB	VISIB_ATTRIBUTES	WDSP	WDSP_ATTRIBUTES	MXSPD	GUST	MAX	MAX_ATTRIBUTES	MIN	MIN_ATTRIBUTES	PRCP	PRCP_ATTRIBUTES	SNPD
FRSHTT	YEAR	Wind_Chill												
[72429793812]	[2017-01-07]	39.106	-84.41609	144.8	[CINCINNATI MUNICI...	[10.5]		24.0	[0.4]	24.0	[1033.5]	24.0	1	
4.3]	24.0	10.0	24.0	7.0	24.0	11.1	[999.9]	16.0		0.0		G	[999.9	
0	[2017]-0.41401565													
[72429793812]	[2017-12-31]	39.106	-84.41609	144.8	[CINCINNATI MUNICI...	[11.0]		24.0	[2.3]	24.0	[1032.4]	22.0	1	
3.4]	24.0	9.6	24.0	5.3]	24.0	9.9	[999.9]	[24.1		0.0		G	[999.9	
1000	[2017] 2.0339768													
[72429793812]	[2017-12-27]	39.106	-84.41609	144.8	[CINCINNATI MUNICI...	[13.0]		24.0	[1.1]	24.0	[1039.3]	23.0	1	
9.9]	24.0	9.5	24.0	5.8]	24.0	9.9	[999.9]	[26.1		0.0		G	[999.9	
1000	[2017] 3.8206456													
[72429793812]	[2017-12-28]	39.106	-84.41609	144.8	[CINCINNATI MUNICI...	[13.6]		24.0	[2.3]	24.0	[1038.8]	24.0	1	
9.4]	24.0	10.0	24.0	5.8]	24.0	8.9	[999.9]	[21.9		0.0		G	[999.9	
0	[2017] 4.533355													
[72429793812]	[2017-01-06]	39.106	-84.41609	144.8	[CINCINNATI MUNICI...	[13.6]		24.0	[4.9]	24.0	[1024.6]	21.0		
5.3]	24.0	10.0	24.0	5.5]	24.0	9.9	[15.9]	[24.1		0.03]		G	[999.9	
0	[2017] 4.868933													
[72429793812]	[2017-01-08]	39.106	-84.41609	144.8	[CINCINNATI MUNICI...	[15.9]		24.0	[5.8]	24.0	[1039.8]	24.0	2	
0.6]	24.0	9.8	24.0	5.2]	24.0	8.9	[999.9]	[23.0		0.0		G	[999.9	
1000	[2017] 7.929748													
[72429793812]	[2017-12-25]	39.106	-84.41609	144.8	[CINCINNATI MUNICI...	[25.8]		24.0	[14.8]	24.0	[1026.6]	19.0		
6.4]	24.0	9.2	24.0	13.5]	24.0	22.0	[32.1]	[34.0		*[0.06]		G	[999.9	
1000	[2017] 14.285113													
[72429793812]	[2017-12-30]	39.106	-84.41609	144.8	[CINCINNATI MUNICI...	[21.6]		24.0	[16.1]	24.0	[1024.3]	17.0		
5.4]	24.0	6.8	24.0	5.3]	24.0	12.0	[20.0]	[28.9		0.07]		G	[999.9	
1000	[2017] 14.539211													
[72429793812]	[2017-01-05]	39.106	-84.41609	144.8	[CINCINNATI MUNICI...	[22.2]		24.0	[15.7]	24.0	[1018.4]	16.0	[99	
9.7]	24.0	6.1	24.0	5.8]	24.0	11.1	[15.9]	[25.0		*[0.0]		G	[999.9	
1000	[2017] 14.748862													
[72429793812]	[2017-12-26]	39.106	-84.41609	144.8	[CINCINNATI MUNICI...	[23.3]		24.0	[10.4]	24.0	[1033.8]	24.0	1	
4.9]	24.0	9.8	24.0	6.2]	24.0	8.9	[999.9]	[28.9		0.0]		G	[999.9	
1000	[2017] 15.688978													

only showing top 10 rows

Predict the maximum Temperature for Cincinnati for November and December 2024, based on the previous 2 years of weather data :

ChromePROJECT 4: Big Data with Py...sairimrlocalhost:8888/notebooks/Desktop/sairimr.ipynb?jupyter Trusted

File Edit View Run Kernel Settings Help

Code

JupyterLabPython 3 (ipykernel)

[49]:

```
from pyspark.ml.regression import LinearRegression
from pyspark.ml.feature import VectorAssembler

cincinnati_data_2023_2024 = full_df.filter(
    (full_df["STATION"] == "72429793812") & (full_df["YEAR"].isin([2023, 2024]))
)

features = ["TEMP", "WDSP", "PRCP", "GUST"]
assembler = VectorAssembler(inputCols=features, outputCol="features")
cincinnati_data_2023_2024 = assembler.transform(cincinnati_data_2023_2024)

lr = LinearRegression(featuresCol="features", labelCol="MAX")
model = lr.fit(cincinnati_data_2023_2024)

# Predict for November and December 2024
predictions = model.transform(cincinnati_data_2023_2024.filter(cincinnati_data_2023_2024["YEAR"] == 2024))
predictions.select("STATION", "NAME", "DATE", "MAX", "prediction").show(2)
```

25/03/23 20:30:48 WARN Instrumentation: [ef407440] regParam is zero, which might cause numerical instability and overfitting.

25/03/23 20:30:48 WARN InstanceBuilder: Failed to load implementation from:dev.ludovic.netlib.blas.JNIBLAS

25/03/23 20:30:48 WARN InstanceBuilder: Failed to load implementation from:dev.ludovic.netlib.lapack.JNILAPACK

STATION	NAME	DATE	MAX	prediction
72429793812	CINCINNATI MUNICI...	2024-01-01	43.0	44.73304454638277
72429793812	CINCINNATI MUNICI...	2024-01-02	39.9	51.07019285302319

only showing top 2 rows

1: