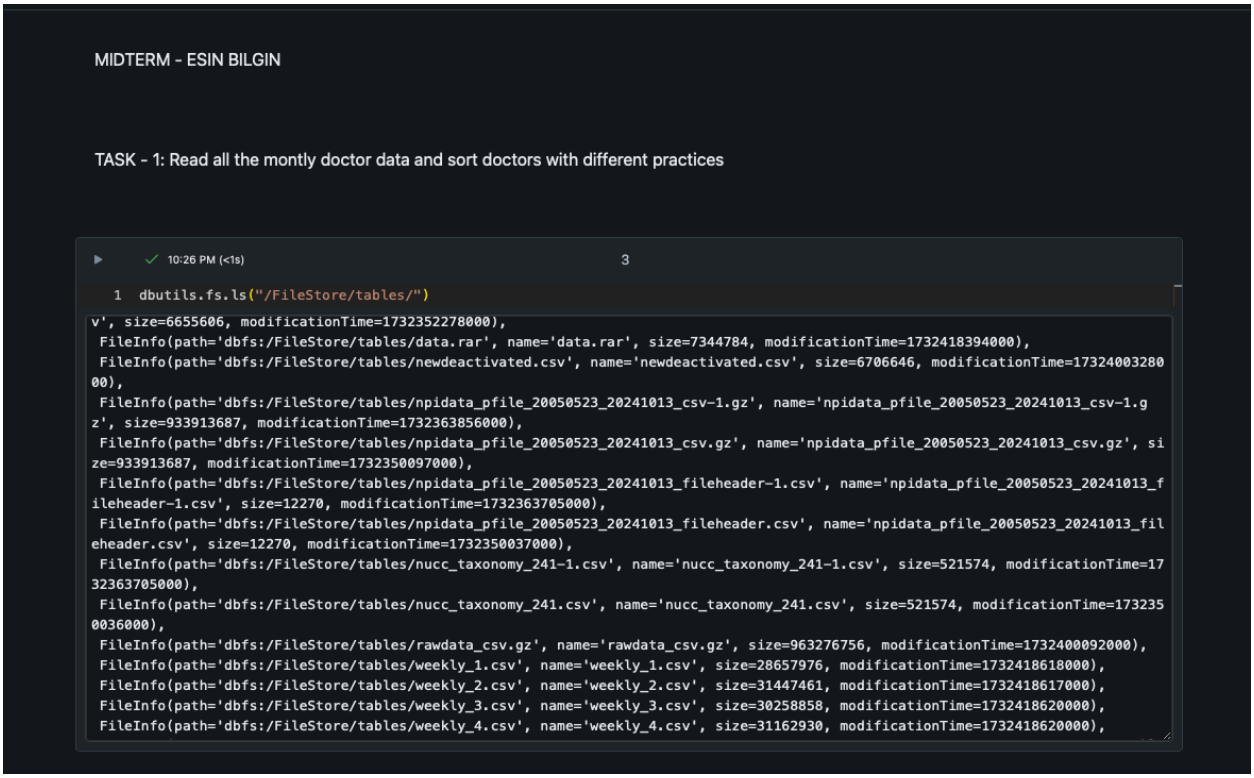
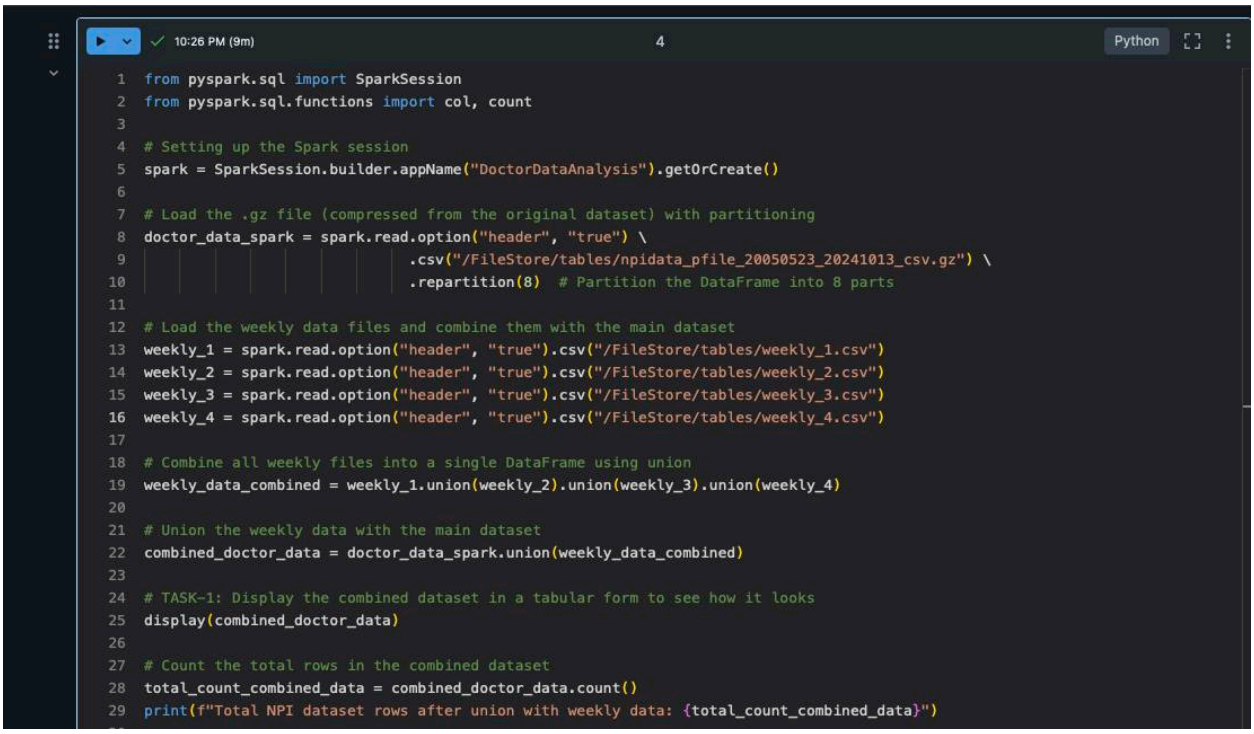
TASK-1

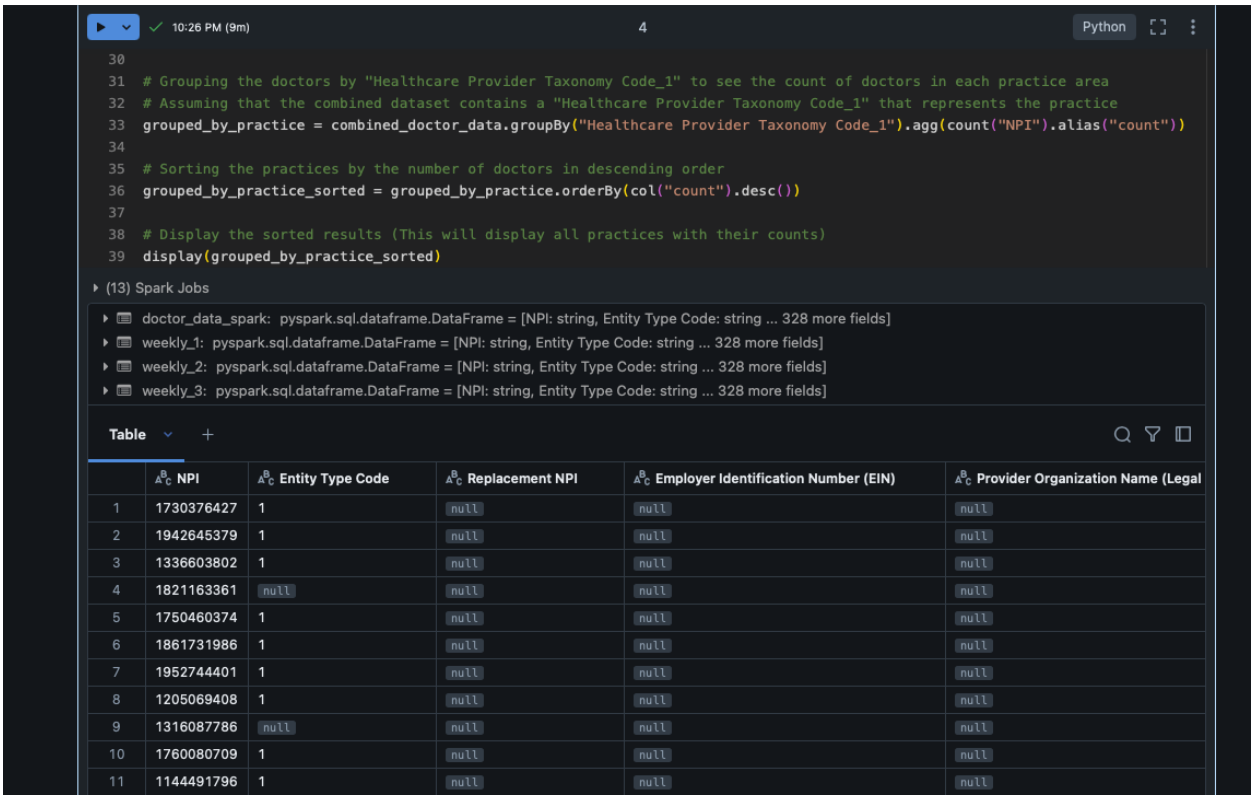
(The first part to follow my tables and all files correctly.)

First, I uploaded the main dataset to Databricks in compressed .gz format and prepared it by dividing it into 8 parts (repartition) for faster processing. After that, I uploaded four separate weekly datasets (weekly_1, weekly_2, weekly_3, weekly_4) and merged them together (union). I combined these weekly data with the main dataset using union again to make one big, combined dataset. Then, I displayed this combined dataset in a table (display()), making sure that the entire data could be examined. Finally, I calculated the total number of rows (count()) in the combined dataset, so that I could clearly see how much data I had because of all these merges. I did these steps in order to analyze the data correctly and comprehensively.

NPI DATASET COUNTS AFTER UNION : 8677330







	NPI	Entity Type Code	Replacement NPI	Employer Identification Number (EIN)	Provider Organization Name (Legal
1	1730376427	1	null	null	null
2	1942645379	1	null	null	null
3	1336603802	1	null	null	null
4	1821163361	null	null	null	null
5	1750460374	1	null	null	null
6	1861731986	1	null	null	null
7	1952744401	1	null	null	null
8	1205069408	1	null	null	null
9	1316087786	null	null	null	null
10	1760080709	1	null	null	null
11	1144491796	1	null	null	null

1,135+ rows | Truncated data due to byte limit | 9.13 minutes runtime — Refreshed 1 minute ago

Total NPI dataset rows after union with weekly data: 8677330

Table

	Healthcare Provider Taxonomy Code_1	count
1	106S00000X	410825
2	390200000X	324825
3	101YM0800X	303071
4	null	291740
5	183500000X	284728
6	1041C0700X	280558
7	225100000X	273786
8	207R00000X	209578
9	363LF0000X	204483
10	207Q00000X	202036
11	235Z00000X	189332
12	1223G0001X	170318
13	163W00000X	161837
14	111N00000X	150631
15	171M00000X	149329

864 rows | 9.13 minutes runtime — Refreshed 1 minute ago

TASK-2

In this step, I first loaded the datasets containing inactive doctors and their classification information. Then I merged these datasets with my main doctor dataset and added the classification information of each doctor. Using left_anti join, I removed the inactive doctors from the dataset and obtained my final dataset consisting of only active doctors. Finally, I checked the result by viewing this cleaned dataset and calculating the total number of active doctors.

Total records before removing deactivated doctors: 8677330

Total records after removing deactivated doctors: 8387308

```python
from pyspark.sql.functions import col, trim

# Load the deactivation dataset
deactivation_data_spark = spark.read.option("header", "true") \
    .csv("/FileStore/tables/NPPES_Deactivated_NPI_Report_20241014.csv")

# Load the taxonomy dataset to enhance the dataset with classifications
taxonomy_data_spark = spark.read.option("header", "true") \
    .csv("/FileStore/tables/nucc_taxonomy_241.csv")

# Trim NPI columns to remove any leading/trailing whitespaces
combined_doctor_data = combined_doctor_data.withColumn("NPI", trim(col("NPI")))
deactivation_data_spark = deactivation_data_spark.withColumn("NPI", trim(col("NPI")))

# Join the combined dataset with the taxonomy data on the taxonomy code
merged_data = combined_doctor_data.join(
    taxonomy_data_spark,
    combined_doctor_data["Healthcare Provider Taxonomy Code_1"] == taxonomy_data_spark["Code"],
    how='left'
)

# Rename the taxonomy classification column to make it more readable
merged_data = merged_data.withColumnRenamed("Display Name", "Practice Name")
```

```python
# Count before removing deactivated doctors
print(f"Total records before removing deactivated doctors: {merged_data.count()}")

# Use left_anti join to keep only active NPIs (remove deactivated doctors)
final_active_data = merged_data.join(
    deactivation_data_spark,
    on="NPI",
    how='left_anti'
)

# Count after removing deactivated doctors
print(f"Total records after removing deactivated doctors: {final_active_data.count()}")

# Display the final DataFrame
display(final_active_data)

total_active_doctor_count = final_active_data.count()
print(f"Total active doctor count: {total_active_doctor_count}")

```

▸ (20) Spark Jobs

- taxonomy_data_spark: pyspark.sql.dataframe.DataFrame = [Code: string, Grouping: string ... 6 more fields]
- combined_doctor_data: pyspark.sql.dataframe.DataFrame = [NPI: string, Entity Type Code: string ... 328 more fields]
- deactivation_data_spark: pyspark.sql.dataframe.DataFrame = [NPI: string, NPPES Deactivation Date: string]
- merged_data: pyspark.sql.dataframe.DataFrame = [NPI: string, Entity Type Code: string ... 336 more fields]
- final_active_data: pyspark.sql.dataframe.DataFrame = [NPI: string, Entity Type Code: string ... 336 more fields]

```
Total records before removing deactivated doctors: 8677330
Total records after removing deactivated doctors: 8387308
```

▸ (20) Spark Jobs

- taxonomy_data_spark: pyspark.sql.dataframe.DataFrame = [Code: string, Grouping: string ... 6 more fields]
- combined_doctor_data: pyspark.sql.dataframe.DataFrame = [NPI: string, Entity Type Code: string ... 328 more fields]
- deactivation_data_spark: pyspark.sql.dataframe.DataFrame = [NPI: string, NPPES Deactivation Date: string]
- merged_data: pyspark.sql.dataframe.DataFrame = [NPI: string, Entity Type Code: string ... 336 more fields]
- final_active_data: pyspark.sql.dataframe.DataFrame = [NPI: string, Entity Type Code: string ... 336 more fields]

```
Total records before removing deactivated doctors: 8677330
Total records after removing deactivated doctors: 8387308
```

Table +

Add filter

		Provider Organization Name (Legal Business Name)	Provider Last Name (Legal Name)	Provider First Name	Provider Middle Name
1		null	DANIELI	ANNA	AGNIESZKA
2		null	HUYNH	VAN	TUNG
3		null	MALOTT	LEIGH ANNE	null
4		null	BOWIE	CHARLES	HAROLD
5		null	WREN	TERESA	MARIE
6		null	TWEEDIE	DONALD	FERGUSON
7		null	RAYMOND	HOLLY	null
8		null	OCAMPO	ADRIANA	MICHELLE
9		null	CRAWFORD	CRISTA	MARIE
10		null	YALAMANCHILI	RAMESH	null
11		null	STONE	DEBRA	LYNN
12		null	ALLISON	RONDA	J
13		null	BILGI	JAGADISH	R
14		null	PERRY	TAMERA	null

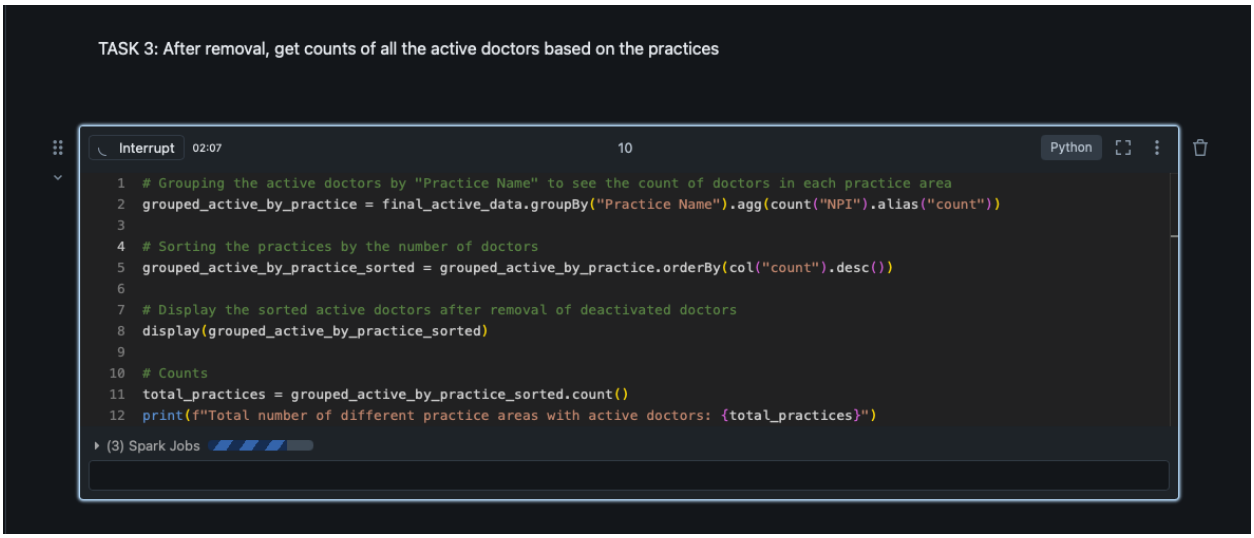
785+ rows | Truncated data due to byte limit | 12.91 minutes runtime

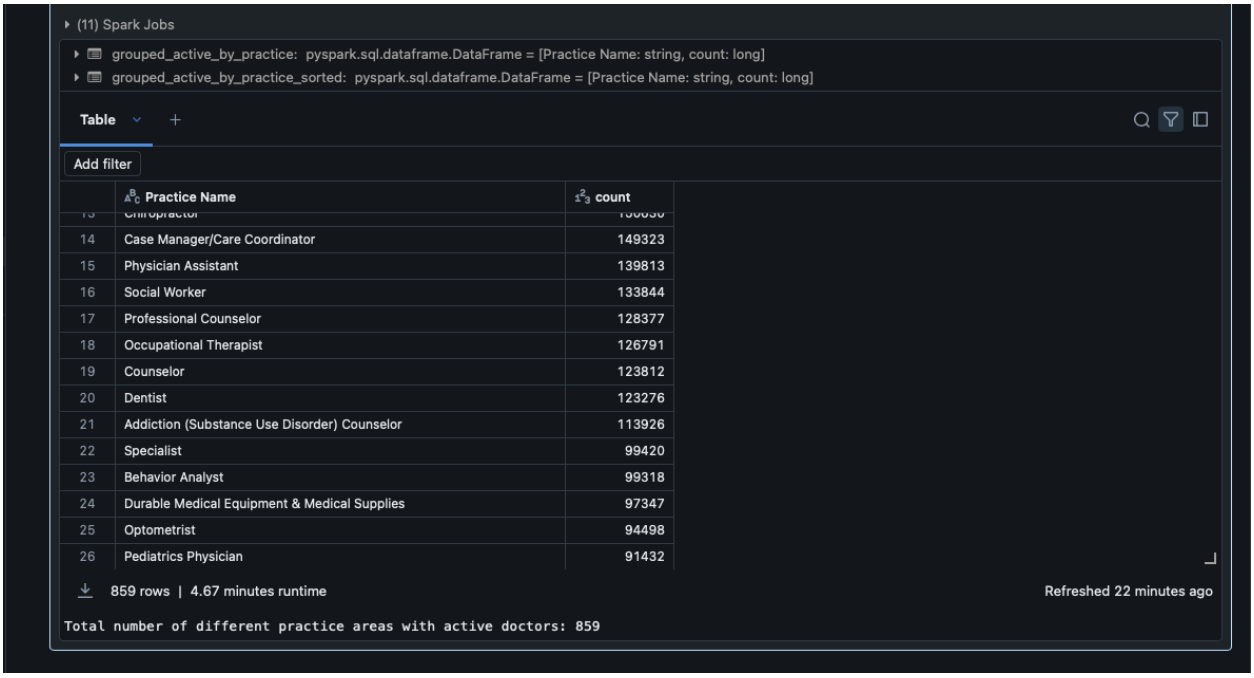
Refreshed 45 minutes ago

```
Total active doctor count: 8387308
```

TASK-3

In this section, I grouped active doctors by practice area and calculated the number of doctors in each practice area. Then I sorted these groups in descending order of the number of doctors. Thanks to this process, I was able to see how many doctors were in each practice area. Finally, I displayed this sorted data and verified it by checking the total number of different practice areas.



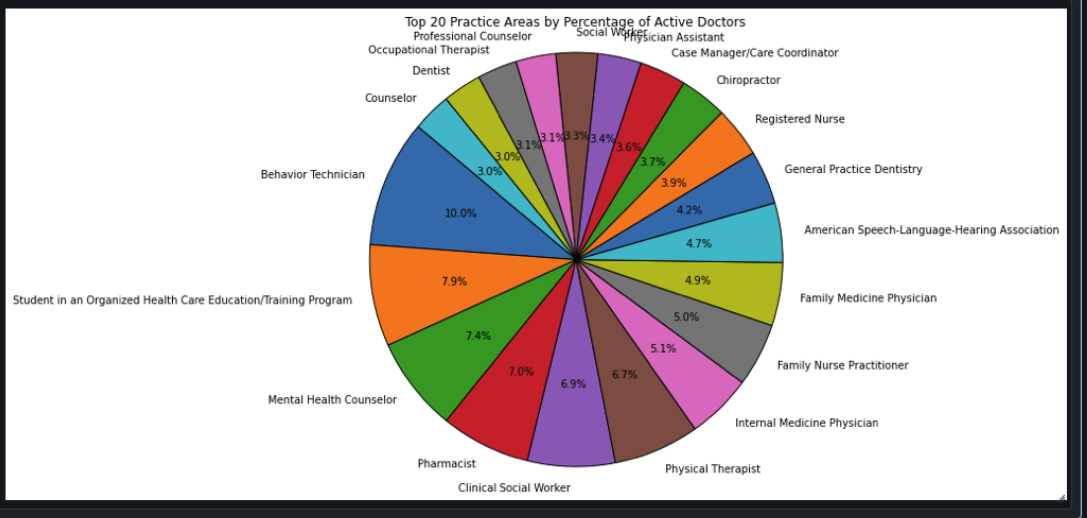


TASK-4

At this stage, I converted the Spark dataset to a Pandas data frame for easier visualization. I stripped out null values so that I could only use valid (non-NULL) practice areas during the visualization. Then, I selected the 20 practice areas with the most active doctors and created a pie chart to show the distribution of these areas. This chart visualized the percentage share of each practice area in the total active doctors, allowing me to more clearly understand which areas were more densely populated.

```python
import pandas as pd
import matplotlib.pyplot as plt

# Converting the Spark to Pandas for visualization
grouped_active_by_practice_pandas = grouped_active_by_practice_sorted.toPandas()

# Dropping null values for visualization
grouped_active_by_practice_pandas_non_null = grouped_active_by_practice_pandas.dropna(subset=['Practice Name'])

# Sorting the data to get top 20 practice areas
top_20_active_practices = grouped_active_by_practice_pandas_non_null.sort_values(by='count', ascending=False).head(20)

# Creating a pie chart
plt.figure(figsize=(10, 8))
plt.pie(
    top_20_active_practices['count'],
    labels=top_20_active_practices['Practice Name'],
    autopct='%1.1f%%',
    startangle=140,
    wedgeprops={'edgecolor': 'black'}
)
plt.title('Top 20 Practice Areas by Percentage of Active Doctors')
plt.axis('equal')  #circle
plt.show()
```

(7) Spark Jobs



TASK – 5

In this piece of code, I am querying my dataset to check if certain doctors exist. I am looking for matches in the dataset based on the doctor's first name, last name, title, and address. If that doctor exists in the dataset, I print the relevant information to the screen. This is an important step to check the accuracy of the data and to understand if certain doctors exist in the datas. While writing this code, I proceeded to check if certain doctors were on the active doctor list. First, I normalized all data inputs (e.g. lowercase all letters, remove unnecessary spaces and punctuation) to ensure that the comparisons were error-free. I searched the large dataset using each doctor's first and last name. If a doctor was on the list, I also checked their credential (e.g. "MD" or "DO") and address/phone information to provide a more specific match. If the credential or address did not match, I improved the code to present the available alternatives for the user to evaluate. In this way, I aimed to provide a detailed infos.

```python
import pandas as pd
import re

# helper function: normalize strings by trimming, converting to lowercase, and removing punctuation
def normalize_string(value):
    return re.sub(r'[^\w\s]', '', value.strip().lower()) if pd.notna(value) else value

# normalize an entire dataframe
def normalize_dataframe(df):
    return df.applymap(normalize_string)

# list of doctors to check
doctors_to_check = [
    {"first_name": "Elizabeth", "last_name": "Botham", "credential": "MD", "address": "1 Seymour St 6th Floor, Montclair, NJ 07042", "phone": None},
    {"first_name": "Michael", "last_name": "Costanzo", "credential": "APN", "address": "890 Mountain Ave 3rd Floor, New Providence, NJ 07974", "phone": None},
    {"first_name": "Jenna", "last_name": "Presto", "credential": "MD", "address": "6 Brighton Rd, Clifton, NJ 07012", "phone": None},
    {"first_name": "Daniel", "last_name": "Roling", "credential": "MD", "address": "6 Lancaster Ave, Wynnewood, PA 19096", "phone": None},
    {"first_name": "Marlyn", "last_name": "Wu", "credential": "DO", "address": "2 Brighton Road, 2nd Floor, Clifton, NJ", "phone": "(973) 792-8455"},
    {"first_name": "Robert", "last_name": "Kantor", "credential": "MD", "address": "1255 Broad St., 2nd Floor, Clifton, NJ", "phone": "(973) 685-5736"}
]

# normalize the list of doctors
doctors_to_check_df = normalize_dataframe(pd.DataFrame(doctors_to_check))
```

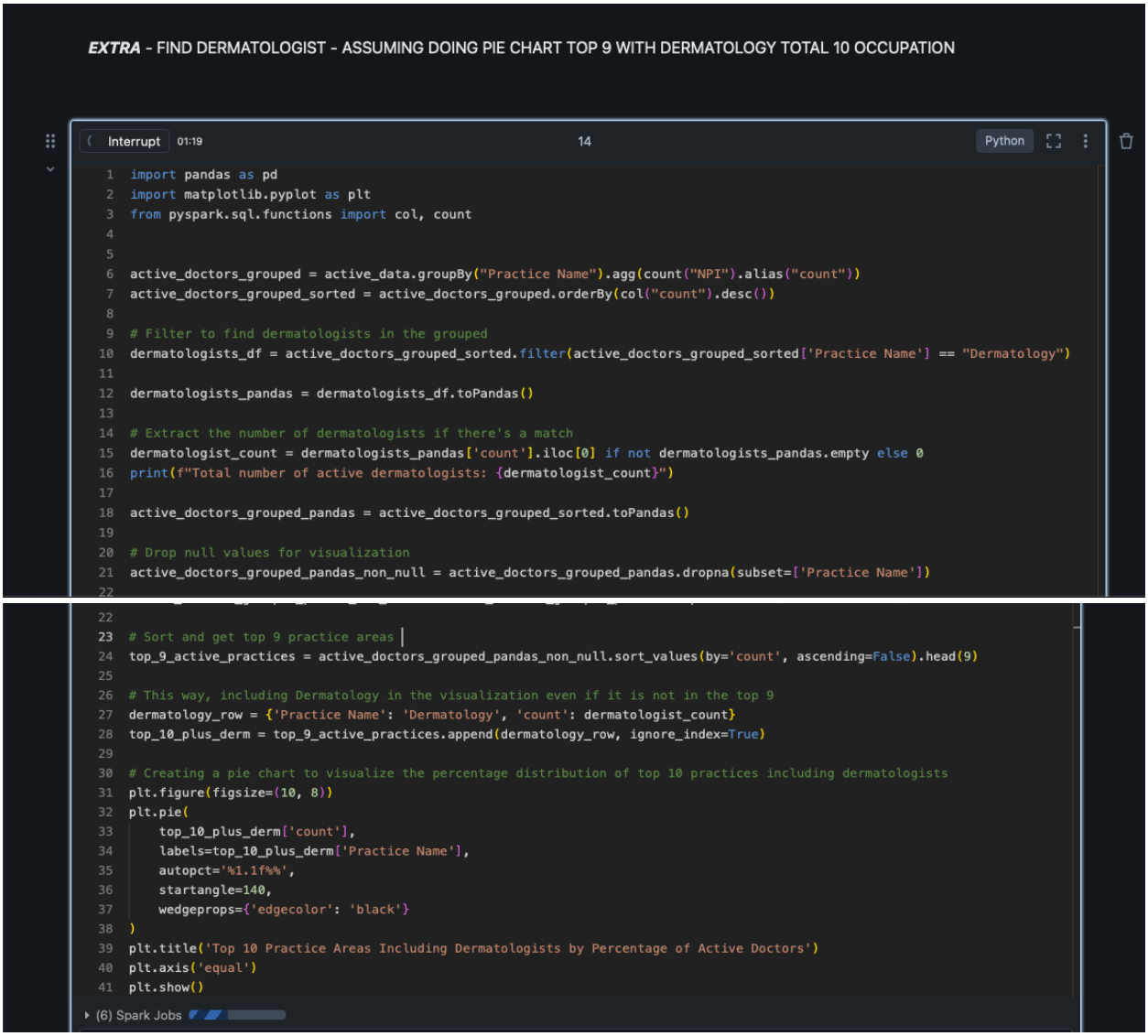
```python
# example active doctor dataset
data = [
    {"Provider First Name": "elizabeth", "Provider Last Name (Legal Name)": "botham", "Provider Credential Text": "MD",
     "Provider First Line Business Practice Location Address": "1 seymour st 6th floor", "Provider Business Practice Location Address City Name": "montclair",
     "Provider Business Practice Location Address State Name": "nj", "Provider Business Practice Location Address Postal Code": "07042",
     "Provider Business Practice Location Address Telephone Number": None},
    {"Provider First Name": "robert", "Provider Last Name (Legal Name)": "kantor", "Provider Credential Text": "MD",
     "Provider First Line Business Practice Location Address": "1255 broad st", "Provider Business Practice Location Address City Name": "clifton",
     "Provider Business Practice Location Address State Name": "nj", "Provider Business Practice Location Address Postal Code": "07013",
     "Provider Business Practice Location Address Telephone Number": "(973) 685-5736"},
    {"Provider First Name": "marlyn", "Provider Last Name (Legal Name)": "wu", "Provider Credential Text": "md",
     "Provider First Line Business Practice Location Address": "2 brighton road", "Provider Business Practice Location Address City Name": "clifton",
     "Provider Business Practice Location Address State Name": "nj", "Provider Business Practice Location Address Postal Code": None,
     "Provider Business Practice Location Address Telephone Number": "(973) 792-8455"}
]
```

```
        # check address and phone matches
        address_match = any(doctor["address"] in str(item) for sublist in all_addresses for item in sublist)
        phone_match = doctor["phone"] and any(doctor["phone"] in phone for phone in all_phones)

        # generate message based on match results
        result_message = f"{doctor['first_name'].title()} {doctor['last_name'].title()} is presented in the list"

        if credential_match and (address_match or phone_match):
            result_message += ", credential matches and address or phone matches."
        elif credential_match:
            result_message += ", credential matches but address and phone do NOT match."
        elif address_match or phone_match:
            result_message += ", address or phone matches but credential does NOT match."
        else:
            result_message += ", but no match on credentials or address/phone."

        # print result message
        print(result_message)

        # display alternative addresses and credentials if full match is not found
        if not (credential_match and (address_match or phone_match)):
            if not credential_match:
                result_message += "\n  Credential options found:"
                for cred in match_found['Provider Credential Text'].unique():
                    print(f"    - {cred}")

            if not address_match or not phone_match:
                if all_addresses:
                    print("  Address options are:")
                    for addr in all_addresses:
                        print(f"    - {' '.join(filter(None, addr))}")
                if all_phones:
                    print("  Phone options are:")
                    for p in all_phones:
                        print(f"    - {p}")

    else:
        # if no match found for the name
        print(f"{doctor['first_name'].title()} {doctor['last_name'].title()}, {doctor['credential']}, is NOT presented in the list.")
```

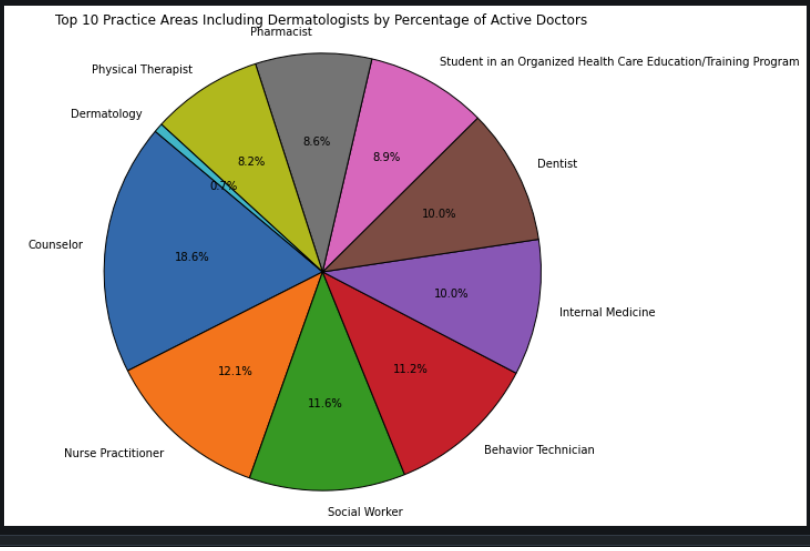
```
Elizabeth Botham is presented in the list, credential matches but address and phone do NOT match.
  Address options are:
    - 1 seymour st 6th floor montclair nj 07042
Michael Costanzo, apn, is NOT presented in the list.
Jenna Presto, md, is NOT presented in the list.
Daniel Roling, md, is NOT presented in the list.
Marlyn Wu is presented in the list, address or phone matches but credential does NOT match.
    - md
  Address options are:
    - 2 brighton road clifton nj
  Phone options are:
    - 973 7928455
Robert Kantor is presented in the list, credential matches and address or phone matches.
```

EXTRA BONUS PART***DUE TO CHANGING***DERMATOLOGY DATA ANALYSIS

IT OBSERVED- TOP 9 OCCUPATION AND DERMATOLOGY



```
Total number of active dermatologists: 24336
<command-3593139635044469>:31: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a future version. Use
pandas.concat instead.
  top_10_plus_derm = top_9_active_practices.append(dermatology_row, ignore_index=True)
```



ADDITIONAL TEST CODES INSIDE OF MY FILE TO CHECK MYSELF IN SOME CASES

CHECKING THE LAST DOCTOR NAME IF DATA LOADED ACCURATELY