**1.**

**df.info():** Gives the overall info of the data frame such as the number of rows, and columns available and their types, any nulls available in the columns, etc.

**plt.figure(figsize=(10,5)):** To adjust the size of the plot so that it is more visually pleasing.

**plt.xticks(rotation=45):** Rotate x-axis labels for better readability.

**df.groupby('Industry')['Mid Salary'].mean().sort\_values(ascending=False)**

Group by industry column and find the mean of Mid Salary for each group and then sort the groups in descending order.

**fil\_df['Company Name'].value\_counts():** To get all unique rows in Company Name and its corresponding count.

**df[df['Location'].str.contains(location)]:** To filter out a particular location from a data frame.

**2.**

1. For better communication through emails, I use Chat GPT to make conversation more formal and articulate my thoughts in a better way.

I take its help to fix the errors in the code I develop like wrong use of functions and others.

Finding if there is a more efficient way for my solution.

1. Yes, the session was helpful in completing the task.
2. The class was very effective. As we have a lab we were able to put our knowledge to test then and there which helps to understand things more efficiently.
3. Yes, it was enjoyable and productive, but it would have been more productive if there were charging points, my laptop ran out of battery in the middle.

3.

#### **Industry Bar Plot**

**User**

plt.bar(df['Industry'].valuecounts())

fix this

**ChatGPT**

It seems like you want to create a bar plot from a DataFrame using Matplotlib, but there is a syntax error in your code. You should use `value\_counts()` to count the occurrences of each unique value in the 'Industry' column and then pass the result to `plt.bar()` for plotting. Here's the corrected code:

```python

import matplotlib.pyplot as plt

# Assuming 'df' is your DataFrame and 'Industry' is the column name

industry\_counts = df['Industry'].value\_counts()

# Plot the bar chart

plt.bar(industry\_counts.index, industry\_counts.values)

# Optionally, you can add labels and titles

plt.xlabel('Industry')

plt.ylabel('Count')

plt.title('Distribution of Industries')

# To make the x-axis labels more readable, you can rotate them if needed

plt.xticks(rotation=90)

# Show the plot

plt.show()

```

This code will create a bar plot showing the distribution of values in the 'Industry' column of your DataFrame. Adjust the labels and titles as needed for your specific use case.

**User**

head(1,6)

**ChatGPT**

It seems like you want to view the first 6 rows of a DataFrame. In Python, you can use the `.head()` method to achieve this. Assuming you have a DataFrame called `df`, you can do the following:

```python

# Display the first 6 rows of the DataFrame

df.head(6)

```

This code will display the first 6 rows of your DataFrame, providing you with a preview of its contents. Adjust the number inside `head()` if you want to view a different number of rows.

**User**

head(2,7)

**ChatGPT**

It looks like you want to view the first 7 rows of a DataFrame, starting from the second row. To achieve this, you can use both the `head()` and `tail()` methods in Pandas. Here's how you can do it:

```python

# Display the first 7 rows of the DataFrame starting from the second row (index 1)

df.iloc[1:8].head(7)

```

This code first selects the rows from index 1 to 7 (excluding the 8th row) using `.iloc[]`, and then it uses `.head(7)` to display the first 7 rows from that selection.

**User**

figsize

**ChatGPT**

The `figsize` parameter is used in Matplotlib to specify the size of a figure (i.e., the overall dimensions of a plot) in inches. You can use this parameter when creating a Matplotlib figure to set its width and height. Here's how you can use `figsize` when creating a Matplotlib figure:

```python

import matplotlib.pyplot as plt

# Define the figsize as (width, height) in inches

fig, ax = plt.subplots(figsize=(8, 6))

# Create your plot or add data to the axes

ax.plot([1, 2, 3, 4], [10, 15, 13, 18])

# Optionally, you can set labels and a title

ax.set\_xlabel('X-axis Label')

ax.set\_ylabel('Y-axis Label')

ax.set\_title('Sample Plot')

# Show the plot

plt.show()

```

In this example, `figsize=(8, 6)` sets the width of the figure to 8 inches and the height to 6 inches. You can adjust these values to customize the size of your Matplotlib figure according to your preferences.

**User**

plt.figsize((10,20))

plt.bar(industry\_count.index,industry\_count.values)

plt.xticks(rotation=45)

plt.show()

**ChatGPT**

It looks like you're trying to set the figure size using `plt.figsize()`, but Matplotlib uses a different method for setting the figure size. You should use `plt.figure(figsize=(width, height))` to specify the figure size. Here's the corrected code:

```python

import matplotlib.pyplot as plt

# Assuming you have already defined industry\_count as a pandas Series

# You can set the figure size here

plt.figure(figsize=(10, 20))

# Create a bar plot

plt.bar(industry\_count.index, industry\_count.values)

# Rotate x-axis labels for better readability

plt.xticks(rotation=45)

# Show the plot

plt.show()

```

This code will create a bar plot with the specified figure size of 10 inches in width and 20 inches in height. Make sure you have the `industry\_count` variable correctly defined before using it in the code.

**User**

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

plt.bar(industry\_count.index,industry\_count.values)

plt.xticks(rotation=45)

plt.xlabel('Industry')

plt.ylabel('Number Of Jobs')

plt.title('Distribution of Industries (Excluding -1)')

plt.show()

make bars thinner

**ChatGPT**

To make the bars thinner in a Matplotlib bar plot, you can adjust the width of the bars by using the `width` parameter when creating the bar plot. Here's how you can modify your code to make the bars thinner:

```python

import matplotlib.pyplot as plt

# Assuming you have already defined industry\_count as a pandas Series

# You can set the figure size here

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

# Create a bar plot with thinner bars by adjusting the width parameter

plt.bar(industry\_count.index, industry\_count.values, width=0.5) # Adjust the width as needed

# Rotate x-axis labels for better readability

plt.xticks(rotation=45)

# Label your axes and add a title

plt.xlabel('Industry')

plt.ylabel('Number Of Jobs')

plt.title('Distribution of Industries (Excluding -1)')

# Show the plot

plt.show()

```

In the code above, I've set the width parameter to `0.5` when creating the bar plot. You can adjust this value to make the bars even thinner or wider according to your preference.

**User**

Rating Company Name Location Headquarters Size Founded Type of ownership Industry Sector Revenue Competitors Easy Apply

3.2 "Vera Institute of Justice

3.2" New York, NY New York, NY 201 to 500 employees 1961 Nonprofit Organization Social Assistance Non-Profit $100 to $500 million (USD) -1 TRUE

3.8 "Visiting Nurse Service of New York

3.8" New York, NY New York, NY 10000+ employees 1893 Nonprofit Organization Health Care Services & Hospitals Health Care $2 to $5 billion (USD) -1 -1

3.4 "Squarespace

3.4" New York, NY New York, NY 1001 to 5000 employees 2003 Company - Private Internet Information Technology Unknown / Non-Applicable GoDaddy -1

4.1 "Celerity

4.1" New York, NY McLean, VA 201 to 500 employees 2002 Subsidiary or Business Segment IT Services Information Technology $50 to $100 million (USD) -1 -1

3.9 "FanDuel

3.9" New York, NY New York, NY 501 to 1000 employees 2009 Company - Private Sports & Recreation Arts, Entertainment & Recreation $100 to $500 million (USD) DraftKings TRUE

3.9 "Point72

3.9" New York, NY Stamford, CT 1001 to 5000 employees 2014 Company - Private Investment Banking & Asset Management Finance Unknown / Non-Applicable -1 -1

4.4 "Two Sigma

4.4" New York, NY New York, NY 1001 to 5000 employees 2001 Company - Private Investment Banking & Asset Management Finance Unknown / Non-Applicable -1 -1

3.7 "GNY Insurance Companies

3.7" New York, NY New York, NY 201 to 500 employees 1914 Company - Private Insurance Carriers Insurance $100 to $500 million (USD) Travelers, Chubb, Crum & Forster TRUE

4 "DMGT

4.0" New York, NY London, United Kingdom 5001 to 10000 employees 1896 Company - Public Venture Capital & Private Equity Finance $1 to $2 billion (USD) Thomson Reuters, Hearst, Pearson -1

4.4 "Riskified

4.4" New York, NY New York, NY 501 to 1000 employees 2013 Company - Private Research & Development Business Services Unknown / Non-Applicable Signifyd, Forter -1

4 "NYU Langone Health

4.0" New York, NY New York, NY 10000+ employees 1841 Hospital Health Care Services & Hospitals Health Care $5 to $10 billion (USD) NewYork-Presbyterian Hospital, Northwell Health -1

-1 BulbHead Fairfield, NJ -1 1 to 50 employees -1 Company - Private -1 -1 Unknown / Non-Applicable -1 -1

3.7 "Montefiore Medical

3.7" New York, NY Bronx, NY 10000+ employees 1884 Nonprofit Organization Health Care Services & Hospitals Health Care $2 to $5 billion (USD) -1 -1

3 "Known

3.0" New York, NY New York, NY 201 to 500 employees 2004 Company - Private Advertising & Marketing Business Services $100 to $500 million (USD) -1 TRUE

3.4 "Advisor Group

3.4" Jersey City, NJ Phoenix, AZ 1001 to 5000 employees 2016 Company - Private Brokerage Services Finance $1 to $5 million (USD) -1 -1

3.6 "CodeGreen Solutions

3.6" New York, NY New York, NY 1 to 50 employees -1 Company - Private Building & Personnel Services Business Services Unknown / Non-Applicable -1 -1

3.8 "Undertone

3.8" New York, NY New York, NY 201 to 500 employees 2001 Subsidiary or Business Segment Advertising & Marketing Business Services Unknown / Non-Applicable DoubleClick, Specific Media, Collective TRUE

3.8 "NYSTEC

3.8" New York, NY Rome, NY 51 to 200 employees 1996 Nonprofit Organization Consulting Business Services $25 to $50 million (USD) KPMG, Accenture, Deloitte -1

3.9 "Education Development Center, Inc.

3.9" New York, NY Waltham, MA 501 to 1000 employees 1958 Nonprofit Organization Research & Development Business Services $100 to $500 million (USD) -1 -1

4.9 "Teachers Pay Teachers

4.9" New York, NY New York, NY 51 to 200 employees 2006 Company - Private Internet Information Technology Unknown / Non-Applicable -1 -1

Create a bar plot of mean salary estimate from the same top 5 industries - see sample output #2: Hint

You will need to first clean the “Salary Estimate” column

You may use the mid salary as the cleaned version of the “Salary Estimate”

**ChatGPT**

To create a bar plot of the mean salary estimate for the top 5 industries based on the cleaned "Salary Estimate" column, you'll need to perform the following steps:

1. Clean the "Salary Estimate" column to extract the mid-salary values.

2. Group the data by the "Industry" column.

3. Calculate the mean salary estimate for each industry.

4. Select the top 5 industries with the highest mean salary estimates.

5. Create a bar plot.

Here's some sample code to help you get started:

```python

import pandas as pd

import matplotlib.pyplot as plt

import re

# Sample data (replace this with your actual data)

data = {

'Industry': ['Social Assistance', 'Health Care Services & Hospitals', 'Internet', 'IT Services', 'Sports & Recreation'],

'Salary Estimate': ['$60K-$80K', '$70K-$90K', '$80K-$100K', '$90K-$110K', '$100K-$120K']

}

df = pd.DataFrame(data)

# Clean the "Salary Estimate" column and extract the mid-salary values

def clean\_salary(salary):

match = re.search(r'\$(\d+)K-\$(\d+)K', salary)

if match:

lower\_salary = int(match.group(1))

upper\_salary = int(match.group(2))

mid\_salary = (lower\_salary + upper\_salary) / 2

return mid\_salary

else:

return None

df['Mid Salary'] = df['Salary Estimate'].apply(clean\_salary)

# Group by "Industry" and calculate the mean salary estimate

industry\_salary\_mean = df.groupby('Industry')['Mid Salary'].mean().sort\_values(ascending=False)

# Select the top 5 industries

top\_5\_industries = industry\_salary\_mean.head(5)

# Create a bar plot

plt.figure(figsize=(10, 6))

top\_5\_industries.plot(kind='bar', color='skyblue')

plt.xlabel('Industry')

plt.ylabel('Mean Salary Estimate')

plt.title('Mean Salary Estimate by Top 5 Industries')

plt.xticks(rotation=45)

# Show the plot

plt.show()

```

In this code, we first clean the "Salary Estimate" column to extract the mid-salary values and then calculate the mean salary estimate for each industry. Finally, we select the top 5 industries with the highest mean salary estimates and create a bar plot to visualize the results.

**User**

df[df['Industry'] in list(industry\_count.index)]

fix this

**ChatGPT**

To filter the DataFrame `df` based on whether the 'Industry' column values are present in the list of industry names from `industry\_count.index`, you can use the `.isin()` method. Here's the corrected code:

```python

# Assuming you have already defined industry\_count as a pandas Series

# and df is your DataFrame

# Filter the DataFrame based on 'Industry' column values

filtered\_df = df[df['Industry'].isin(industry\_count.index.tolist())]

# Now, filtered\_df contains only the rows where 'Industry' matches the list of industry names

# You can use filtered\_df for further analysis or visualization

```

This code will create a new DataFrame `filtered\_df` that contains only the rows where the 'Industry' column values are present in the list of industry names from `industry\_count.index`.

**User**

# Clean the "Salary Estimate" column and extract the mid-salary values

def clean\_salary(salary):

match = re.search(r'\$(\d+)K-\$(\d+)K', salary)

if match:

lower\_salary = int(match.group(1))

upper\_salary = int(match.group(2))

mid\_salary = (lower\_salary + upper\_salary) / 2

return mid\_salary

else:

return None

#filtering only rows in top 5 industries

df=df[df['Industry'].isin(industry\_count.index.tolist())]

df['Mid Salary'] = df['Salary Estimate'].apply(clean\_salary)

# Group by "Industry" and calculate the mean salary estimate

industry\_salary\_mean = df.groupby('Industry')['Mid Salary'].mean().sort\_values(ascending=False)

# Select the top 5 industries

top\_5\_industries = industry\_salary\_mean.head(5)

# Creating a bar plot

plt.figure(figsize=(10, 6))

top\_5\_industries.plot(kind='bar', color='skyblue')

plt.xlabel('Industry')

plt.ylabel('Mean Salary Estimate')

plt.title('Mean Salary Estimate by Top 5 Industries')

plt.xticks(rotation=45)

plt.show()

remove y axis labels and put them on bars

**ChatGPT**

To remove the y-axis labels and place them on the bars in a horizontal bar plot, you can modify the code as follows:

```python

import matplotlib.pyplot as plt

import pandas as pd

import re

# Sample data (replace this with your actual data)

data = {

'Industry': ['Social Assistance', 'Health Care Services & Hospitals', 'Internet', 'IT Services', 'Sports & Recreation'],

'Salary Estimate': ['$60K-$80K', '$70K-$90K', '$80K-$100K', '$90K-$110K', '$100K-$120K']

}

df = pd.DataFrame(data)

# Clean the "Salary Estimate" column and extract the mid-salary values

def clean\_salary(salary):

match = re.search(r'\$(\d+)K-\$(\d+)K', salary)

if match:

lower\_salary = int(match.group(1))

upper\_salary = int(match.group(2))

mid\_salary = (lower\_salary + upper\_salary) / 2

return mid\_salary

else:

return None

df['Mid Salary'] = df['Salary Estimate'].apply(clean\_salary)

# Group by "Industry" and calculate the mean salary estimate

industry\_salary\_mean = df.groupby('Industry')['Mid Salary'].mean().sort\_values(ascending=False)

# Select the top 5 industries

top\_5\_industries = industry\_salary\_mean.head(5)

# Creating a horizontal bar plot

plt.figure(figsize=(10, 6))

ax = top\_5\_industries.plot(kind='barh', color='skyblue')

ax.invert\_yaxis() # Invert the y-axis to have the highest salary at the top

plt.xlabel('Mean Salary Estimate')

plt.title('Mean Salary Estimate by Top 5 Industries')

# Add y-axis labels on the bars

for i, v in enumerate(top\_5\_industries):

ax.text(v + 0.1, i, f'{v:.2f}', va='center', fontsize=10, color='black')

plt.show()

```

In this modified code, we use `ax.text()` to add y-axis labels on the bars, and `ax.invert\_yaxis()` is used to invert the y-axis so that the highest salary is at the top. This results in a horizontal bar plot with labels on the bars.

**User**

give code to plot job title with highest average rating

**ChatGPT**

To plot the job titles with the highest average rating, you can follow these steps:

1. Group the DataFrame by job title.

2. Calculate the average rating for each job title.

3. Select the job titles with the highest average ratings.

4. Create a bar plot to visualize the data.

Here's the code to achieve this:

```python

import pandas as pd

import matplotlib.pyplot as plt

# Sample data (replace this with your actual data)

data = {

'Job Title': ['Data Scientist', 'Software Engineer', 'Product Manager', 'Data Analyst', 'Machine Learning Engineer'],

'Rating': [4.2, 4.5, 4.1, 4.3, 4.4]

}

df = pd.DataFrame(data)

# Group by "Job Title" and calculate the average rating

job\_title\_avg\_rating = df.groupby('Job Title')['Rating'].mean().sort\_values(ascending=False)

# Select the job titles with the highest average ratings

top\_job\_titles = job\_title\_avg\_rating.head()

# Create a bar plot

plt.figure(figsize=(10, 6))

top\_job\_titles.plot(kind='bar', color='skyblue')

plt.xlabel('Job Title')

plt.ylabel('Average Rating')

plt.title('Job Titles with Highest Average Ratings')

# Show the plot

plt.show()

```

In this code, we first group the DataFrame by 'Job Title' and calculate the average rating for each title. Then, we select the job titles with the highest average ratings and create a bar plot to visualize the results.

**User**

top 5 companies that are easy to apply

**ChatGPT**

To find the top 5 companies that are easy to apply to, you can filter the DataFrame based on the "Easy Apply" column and select the companies with a value of `True`. Here's the code to do that:

```python

import pandas as pd

# Sample data (replace this with your actual data)

data = {

'Company Name': ['Company A', 'Company B', 'Company C', 'Company D', 'Company E'],

'Easy Apply': [True, True, False, True, True]

}

df = pd.DataFrame(data)

# Filter the DataFrame to select companies with Easy Apply = True

easy\_to\_apply\_companies = df[df['Easy Apply']]

# Select the top 5 companies

top\_5\_easy\_to\_apply\_companies = easy\_to\_apply\_companies.head(5)

# Display the top 5 companies

print(top\_5\_easy\_to\_apply\_companies)

```

Replace the sample data with your actual DataFrame that contains the company names and the "Easy Apply" column. The `easy\_to\_apply\_companies` DataFrame will contain only the rows where "Easy Apply" is `True`, and `top\_5\_easy\_to\_apply\_companies` will contain the top 5 companies that are easy to apply to.

**User**

exclude rows with -1 in easy apply column

**ChatGPT**

To exclude rows with a value of `-1` in the "Easy Apply" column, you can filter the DataFrame to select only the rows where "Easy Apply" is `True`. Here's the code to do that:

```python

import pandas as pd

# Sample data (replace this with your actual data)

data = {

'Company Name': ['Company A', 'Company B', 'Company C', 'Company D', 'Company E'],

'Easy Apply': [True, True, False, True, -1]

}

df = pd.DataFrame(data)

# Filter the DataFrame to select rows where Easy Apply is not -1

filtered\_df = df[df['Easy Apply'] != -1]

# Display the resulting DataFrame

print(filtered\_df)

```

Replace the sample data with your actual DataFrame that contains the company names and the "Easy Apply" column. The `filtered\_df` DataFrame will contain only the rows where "Easy Apply" is not equal to `-1`.

**User**

# Filter the DataFrame to select companies with Easy Apply = True and Easy Apply !=-1

easy\_to\_apply\_companies = df[df['Easy Apply'] == True and df['Easy Apply'] != -1]

# Select the top 5 companies

top\_5\_easy\_to\_apply\_companies = easy\_to\_apply\_companies.head(5)

# Display the top 5 companies

print(top\_5\_easy\_to\_apply\_companies)

fix this

**ChatGPT**

You can't use the `and` operator directly within the DataFrame filtering condition. Instead, you should use the `&` operator for element-wise logical AND operations within the DataFrame. Here's the corrected code:

```python

import pandas as pd

# Sample data (replace this with your actual data)

data = {

'Company Name': ['Company A', 'Company B', 'Company C', 'Company D', 'Company E'],

'Easy Apply': [True, True, False, True, -1]

}

df = pd.DataFrame(data)

# Filter the DataFrame to select companies with Easy Apply = True and Easy Apply != -1

easy\_to\_apply\_companies = df[(df['Easy Apply'] == True) & (df['Easy Apply'] != -1)]

# Select the top 5 companies

top\_5\_easy\_to\_apply\_companies = easy\_to\_apply\_companies.head(5)

# Display the top 5 companies

print(top\_5\_easy\_to\_apply\_companies)

```

In this corrected code, we use `&` to perform an element-wise logical AND operation to filter rows where "Easy Apply" is `True` and not equal to `-1`.

**User**

# Filter the DataFrame to select companies with Easy Apply = True and Easy Apply !=-1

easy\_to\_apply\_companies = df[df['Easy Apply'] == True]['Easy Apply' != -1]

# Select the top 5 companies

top\_5\_easy\_to\_apply\_companies = easy\_to\_apply\_companies.head(5)

# Display the top 5 companies

print(top\_5\_easy\_to\_apply\_companies)

fix this

**ChatGPT**

To filter the DataFrame to select companies with "Easy Apply" equal to `True` and exclude rows with `-1` in the "Easy Apply" column, you can do the following:

```python

import pandas as pd

# Sample data (replace this with your actual data)

data = {

'Company Name': ['Company A', 'Company B', 'Company C', 'Company D', 'Company E'],

'Easy Apply': [True, True, False, True, -1]

}

df = pd.DataFrame(data)

# Filter the DataFrame to select companies with Easy Apply = True and Easy Apply != -1

easy\_to\_apply\_companies = df[(df['Easy Apply'] == True) & (df['Easy Apply'] != -1)]

# Select the top 5 companies

top\_5\_easy\_to\_apply\_companies = easy\_to\_apply\_companies.head(5)

# Display the top 5 companies

print(top\_5\_easy\_to\_apply\_companies)

```

This code will filter the DataFrame to include only rows where "Easy Apply" is equal to `True` and not equal to `-1`, and then it selects the top 5 companies that meet these criteria.

**User**

change condition so that only rows with Easy Apply True are included

then on that data frame remove any -1 in same column

**ChatGPT**

To filter the DataFrame to include only rows where "Easy Apply" is equal to `True` and then remove any `-1` values in the same column, you can use the following code:

```python

import pandas as pd

# Sample data (replace this with your actual data)

data = {

'Company Name': ['Company A', 'Company B', 'Company C', 'Company D', 'Company E'],

'Easy Apply': [True, True, False, True, -1]

}

df = pd.DataFrame(data)

# Filter the DataFrame to select rows where Easy Apply is True

easy\_to\_apply\_companies = df[df['Easy Apply'] == True]

# Remove rows where Easy Apply is -1 in the filtered DataFrame

easy\_to\_apply\_companies = easy\_to\_apply\_companies[easy\_to\_apply\_companies['Easy Apply'] != -1]

# Display the resulting DataFrame

print(easy\_to\_apply\_companies)

```

In this code, we first filter the DataFrame to include only rows where "Easy Apply" is `True`. Then, we use another filtering step to remove any rows where "Easy Apply" is `-1` within the filtered DataFrame. The resulting `easy\_to\_apply\_companies` DataFrame will contain only the rows with "Easy Apply" as `True` and no `-1` values in the same column.

**User**

df['Easy Apply'].unique()

**ChatGPT**

The `df['Easy Apply'].unique()` code will return an array of unique values in the "Easy Apply" column of your DataFrame. This can be useful to see what distinct values exist in that column. Typically, you would use this to check the variety of values and see if there are any unexpected or unusual entries. Here's how you can use it:

```python

import pandas as pd

# Sample data (replace this with your actual data)

data = {

'Company Name': ['Company A', 'Company B', 'Company C', 'Company D', 'Company E'],

'Easy Apply': [True, True, False, True, -1]

}

df = pd.DataFrame(data)

# Get unique values in the 'Easy Apply' column

unique\_values = df['Easy Apply'].unique()

# Print the unique values

print(unique\_values)

```

In this example, you'll see that the unique values are `[True, False, -1]`, which represents the different values found in the "Easy Apply" column of your DataFrame.

**User**

fil\_df.groupby('Easy Apply').count().sort(ascending=False).head(5)

fix this

**ChatGPT**

It appears that you want to group your DataFrame, `fil\_df`, by the 'Easy Apply' column, count the occurrences of each value, and then sort the result in descending order to get the top 5 counts. You can achieve this using the following corrected code:

```python

# Group by 'Easy Apply', count occurrences, and sort in descending order

top\_5\_easy\_apply\_counts = fil\_df['Easy Apply'].value\_counts().sort\_values(ascending=False).head(5)

# Display the top 5 counts

print(top\_5\_easy\_apply\_counts)

```

This code will give you the top 5 counts of different values in the 'Easy Apply' column of your DataFrame, sorted in descending order.

**User**

fil\_df.groupby('Company Name').count().sort(ascending=False).head(5)

fix this

**ChatGPT**

It looks like you want to group your DataFrame `fil\_df` by the 'Company Name' column, count the occurrences of each company name, and then sort the result in descending order to get the top 5 companies with the highest counts. Here's the corrected code:

```python

# Group by 'Company Name', count occurrences, and sort in descending order

top\_5\_company\_counts = fil\_df['Company Name'].value\_counts().sort\_values(ascending=False).head(5)

# Display the top 5 counts

print(top\_5\_company\_counts)

```

This code will give you the top 5 companies with the highest counts in the 'Company Name' column of your DataFrame, sorted in descending order based on the counts.