Chat with AI for project three

Entry #1

Can you teach me a possible pd function that helps create a cut off date for a particular data frame

###### **ChatGPT said:**

Yes! Pandas has simple and effective ways to create a **cutoff date** using pd.to\_datetime() and filtering with comparison operators.

### **✅ Example using pd.to\_datetime() and filtering:**

python

CopyEdit

df['date'] = pd.to\_datetime(df[['year', 'month', 'day']])

cutoff\_date = pd.to\_datetime("2004-06-22")

df = df[df['date'] >= cutoff\_date]

Entry#2

how can i make my histogram more aesthetically pleasing this is the code so far fig\_hist = px.bar(article\_counts, x='date', y='article\_count', title=' Number of Articles Per Day (Post Sep 7, 2023)', labels={'date': 'Date', 'article\_count': 'Number of Articles'})

###### **ChatGPT said:**

Great! You're off to a solid start — now let's make your histogram look **cleaner and more aesthetically pleasing** using a few tweaks with Plotly Express (px.bar).

### **✅ Here's an improved version:**

python

CopyEdit

import plotly.express as px

fig\_hist = px.bar(

article\_counts,

x='date',

y='article\_count',

title='📊 Number of Articles Per Day (Since Sep 7, 2023)',

labels={'date': 'Date', 'article\_count': 'Articles'},

color='article\_count', # adds a color gradient

color\_continuous\_scale='Viridis' # nice color scheme

)

Entry#3

Is there a possible technique to alter the histogram in such a way where the date has a corresponding day for example sep 7 2023 was a thursday

###### **ChatGPT said:**

Great next step! You can easily **add the day of the week** to your histogram by creating a new column in your article\_counts DataFrame that shows the **day name** (like “Thursday”).

### **✅ Here's how to do it step by step:**

Add this line **right after** you create article\_counts:

python

CopyEdit

article\_counts['day'] = article\_counts['date'].dt.day\_name()

This uses .dt.day\_name() to extract the full weekday name from each date.

### **💡 Then update your x-axis to show both date and day:**

You can combine the date and day name into a new column for the x-axis:

python

CopyEdit

article\_counts['date\_day'] = article\_counts['date'].dt.strftime('%b %d, %Y') + ' (' + article\_counts['day'] + ')'

This makes the label look like:  
 **"Sep 07, 2023 (Thursday)"**

Entry #4

similar to how i made this code article\_counts = df.groupby('date').size().reset\_index(name='article\_count') article\_counts['day'] = article\_counts['date'].dt.day\_name() article\_counts['date\_day'] = article\_counts['date'].dt.strftime('%b %d, %Y') + ' (' + article\_counts['day'] + ')' can i do the same for weekdays

###### **ChatGPT said:**

Yes, absolutely — you can do **something very similar** for weekdays! Here's how you can adapt your logic to build a nice, labeled column for weekdays with article counts.

### **✅ What You Already Did:**

You combined the **date** with the **day name** like this:

python

CopyEdit

article\_counts['date\_day'] = article\_counts['date'].dt.strftime('%b %d, %Y') + ' (' + article\_counts['day'] + ')'

### **✅ Now For Weekdays:**

If you're grouping by **weekday**, and you want a column like "Monday (X articles)", here’s a simple way to do it:

python

CopyEdit

# Step 1: Add weekday names

df['weekday'] = df['date'].dt.day\_name()

# Step 2: Count number of articles per weekday

weekday\_counts = df['weekday'].value\_counts().reindex(

['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday']

).reset\_index()

weekday\_counts.columns = ['Weekday', 'Article Count']

# Step 3: Create combined label like: "Monday (12 articles)"

weekday\_counts['label'] = weekday\_counts['Weekday'] + ' (' + weekday\_counts['Article Count'].astype(str) + ' articles)'

can u explain what each line of code does

###### **ChatGPT said:**

Absolutely! Let's walk through each line of this code **step by step** so you understand exactly what it does:

python

CopyEdit

# Step 1: Add weekday names

df['weekday'] = df['date'].dt.day\_name()

✅ **What it does**:

* Looks at the date column in your DataFrame.
* Uses .dt.day\_name() to get the **full name of the weekday** (like "Monday", "Tuesday").
* Saves that into a new column called 'weekday'.

python

CopyEdit

# Step 2: Count number of articles per weekday

weekday\_counts = df['weekday'].value\_counts().reindex(

['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday']

).reset\_index()

✅ **What it does**:

* df['weekday'].value\_counts() counts how many times each weekday appears (i.e., how many articles on each day).
* reindex([...]) puts the weekdays in **correct order** (instead of random order based on frequency).
* .reset\_index() turns the result into a **clean DataFrame** with numbered rows.

python

CopyEdit

weekday\_counts.columns = ['Weekday', 'Article Count']

✅ **What it does**:

* Renames the default column names (index and weekday) to more meaningful ones:  
  + 'Weekday': Day name (e.g. "Monday")
  + 'Article Count': Number of articles on that day

python

CopyEdit

# Step 3: Create combined label like: "Monday (12 articles)"

weekday\_counts['label'] = weekday\_counts['Weekday'] + ' (' + weekday\_counts['Article Count'].astype(str) + ' articles)'

✅ **What it does**:

* Creates a **new column** called 'label' that combines the weekday and the article count.
* Uses string concatenation to make a readable label like: "Monday (12 articles)".
* .astype(str) is used to convert the number to a string, so it can be joined with text.

### **🧠 Summary:**

| **Line** | **Purpose** |
| --- | --- |
| 1 | Add weekday names like "Monday" to each row |
| 2 | Count how many articles were published on each weekday |
| 3 | Rename columns for clarity |
| 4 | Create a custom label for each weekday like "Tuesday (14 articles)" |

Let me know if you want to use this label column in a chart or for display!

Traceback (most recent call last): File "/Users/faiz/Downloads/FASDH25-portfolio3/lengths\_faiz\_ahmed.py", line 16, in <module> df2['date'] = pd.to\_datetime(df2[['year', 'month','day']]) File "/Library/Frameworks/Python.framework/Versions/3.13/lib/python3.13/site-packages/pandas/core/frame.py", line 4108, in \_\_getitem\_\_ indexer = self.columns.\_get\_indexer\_strict(key, "columns")[1] File "/Library/Frameworks/Python.framework/Versions/3.13/lib/python3.13/site-packages/pandas/core/indexes/base.py", line 6200, in \_get\_indexer\_strict self.\_raise\_if\_missing(keyarr, indexer, axis\_name) File "/Library/Frameworks/Python.framework/Versions/3.13/lib/python3.13/site-packages/pandas/core/indexes/base.py", line 6252, in \_raise\_if\_missing raise KeyError(f"{not\_found} not in index") KeyError: "['day'] not in index" i am recieving this error can you guide me through what i might be doing wrong

###### **ChatGPT said:**

You're seeing this error because the DataFrame df2 **does not contain a column named 'day'**, but your code tries to use it here:

python

CopyEdit

df2['date'] = pd.to\_datetime(df2[['year', 'month','day']])

### **🧠 What the error means:**

* KeyError: "['day'] not in index" means that pandas is trying to find a 'day' column in df2, but it doesn't exist.
* This often happens if your CSV only has columns like year, month, and maybe length\_mean, **but not day**.

### **✅ How to fix it:**

If you're trying to convert year and month into a proper date (e.g. for monthly averages), you can safely **set the day to 1 manually** like this:

python

CopyEdit

df2['date'] = pd.to\_datetime(df2[['year', 'month']].assign(day=1))

This tells pandas: "Use the year and month from the file, and just assume the day is 1."