Exploring Pandas: Common Data Operations

Welcome to this Jupyter Notebook! 🚀 In this notebook, you'll practice some of the most commonly used operations in the **Pandas** library using **two datasets**:

1. **../../data/students.csv** (CSV)
2. **../../data/enrollments.json** (JSON)

These files should be placed in the same folder as this notebook. By the end, you'll have a strong grasp of common data manipulation tasks, and you'll even merge these two datasets on a common key.

Before starting, make sure you have **Pandas** installed. (It should come preinstalled in Anaconda!)

If pandas is not installed, follow the instructions below.

Checking if Pandas is Installed in Your Conda Environment

Before proceeding, check if Pandas is installed in your Conda environment by running the following command in a **Jupyter Notebook** cell:

In [33]:
```
import pandas as pd
print(pd.__version__)
```

```
2.0.3
```

If this runs without errors and prints a version number, Pandas is installed. If you see an **ImportError**, install Pandas using one of the following methods:

For Conda Users (Recommended)

Run this in your terminal or Anaconda Prompt:

```
conda install pandas
```

Using Conda-Forge (If Needed)

If you encounter issues, you can install Pandas from **Conda-Forge**, a community-maintained repository with up-to-date packages:

```
conda install -c conda-forge pandas
```

For Pip Users

If you're using a virtual environment outside Conda, install Pandas via Pip:

```
pip install pandas
```

Now, let’s dive in! 🏊

---

1. Load a CSV file into a Pandas DataFrame

First, let's **import Pandas** and load the datasets. Two datasets have been prepared for you:

- `students.csv`
- `enrollments.json`

You will use these two datasets for the following challenges.

💡 **Hint:** If the file is in the same directory as your notebook, you can just use the filename. Otherwise, provide the full file path.

In [41]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [43]:
```
# Load students.csv
students_df = pd.read_csv("../../data/students.csv")

# Load enrollments.json
enrollments_df = pd.read_json("../../data/enrollments.json")

# Display the first few rows of both datasets
print(students_df.head())
print(enrollments_df.head())
```

```
  student_id First_Name last_name  Birthddate gender        majorField  \
0     STU001       John       Doe   4/12/1998      M  Computer Science   
1     STU002      Maria  Gonzalez    9/5/1997      F           Biology   
2     STU003      Priya     Patel   1/23/1999      F       Engineering   
3     STU004       Alex   Johnson  12/15/1996      M       Mathematics   
4     STU005      Emily     Smith   7/30/2000      F           Physics   

  admission_year  current gpa               contact_email  mobile number  \
0           2020          3.5        john.doe@example.com          -655.0   
1           2019          3.8  maria.gonzalez@example.com          -656.0   
2           2021          3.7      priya.patel@example.com          -657.0   
3           2018          3.2     alex.johnson@example.com             NaN   
4           2022          3.9      emily.smith@example.com          -659.0   

  home_city HOME COUNTRY  
0     Tampa          USA  
1    London           UK  
2    Mumbai        India  
3    Sydney    Australia  
4  New York          USA  
             enrollment_id stud_ref_id subject_code        course_title  \
0  ENR-STU001-MATH101-AB12      STU001      MATH101          Calculus I   
1   ENR-STU002-ENG201-CD34      STU002       ENG201  English Literature   
2    ENR-STU003-CS301-EF56      STU003        CS301     Programming 101   
3  ENR-STU004-HIST105-GH78      STU004      HIST105       World History   
4  ENR-STU005-CHEM110-IJ90      STU005      CHEM110   Organic Chemistry   

  instructor_name  enroll_count term_offered course_fee final_result  \
0       Dr. Smith            25  Spring 2026       1000            A   
1     Prof. Brown            26    Fall 2025       1100            B   
2      Dr. Taylor            27  Spring 2026       1200           C+   
3         Dr. Lee            28    Fall 2025       1300           B+   
4       Dr. Kumar            29  Spring 2026       1400           B-   

   attend_percentage date_enrolled  
0                 71    2026/01/15  
1                 72    2025-09-01  
2                 73    2026/01/15  
3                 74    2025/09/01  
4                 75    2026-01-15  
```

In [ ]:
```
```

In [3]:
```
# your code here
```

2. View the First and Last Few Rows of Each DataFrame

Check out how your data looks. One method previews the first few records, while another method previews the last few. You can specify the number of rows you want to see by explicitly passing an integer argument.

📝 **Tip:** This is a great time to confirm that columns loaded correctly and to spot any obvious data issues (strange values, mismatched columns, etc.).

In [45]:
```
print(students_df.head())
print(enrollments_df.head())
```

```
  student_id First_Name last_name  Birthddate gender       majorField  \
0     STU001       John       Doe   4/12/1998      M  Computer Science
1     STU002      Maria  Gonzalez    9/5/1997      F          Biology
2     STU003      Priya     Patel   1/23/1999      F      Engineering
3     STU004       Alex   Johnson  12/15/1996      M      Mathematics
4     STU005      Emily     Smith   7/30/2000      F          Physics

  admission_year  current gpa              contact_email  mobile number  \
0           2020          3.5        john.doe@example.com         -655.0
1           2019          3.8  maria.gonzalez@example.com         -656.0
2           2021          3.7     priya.patel@example.com         -657.0
3           2018          3.2    alex.johnson@example.com            NaN
4           2022          3.9     emily.smith@example.com         -659.0

  home_city HOME COUNTRY
0     Tampa          USA
1    London           UK
2    Mumbai        India
3    Sydney    Australia
4  New York          USA
             enrollment_id stud_ref_id subject_code        course_title  \
0  ENR-STU001-MATH101-AB12      STU001      MATH101          Calculus I
1   ENR-STU002-ENG201-CD34      STU002       ENG201  English Literature
2    ENR-STU003-CS301-EF56      STU003        CS301     Programming 101
3  ENR-STU004-HIST105-GH78      STU004      HIST105       World History
4  ENR-STU005-CHEM110-IJ90      STU005      CHEM110   Organic Chemistry

  instructor_name  enroll_count term_offered course_fee final_result  \
0       Dr. Smith            25  Spring 2026       1000            A
1     Prof. Brown            26    Fall 2025       1100            B
2       Dr. Taylor           27  Spring 2026       1200           C+
3          Dr. Lee           28    Fall 2025       1300           B+
4       Dr. Kumar            29  Spring 2026       1400           B-

   attend_percentage date_enrolled
0                 71    2026/01/15
1                 72    2025-09-01
2                 73    2026/01/15
3                 74    2025/09/01
4                 75    2026-01-15
```

In [47]:
```
print(students_df.tail())
print(enrollments_df.tail())
```

```
    student_id First_Name last_name Birthddate gender   majorField  \
98      STU096   Victoria     Ortiz   8/2/1996      F  Mathematics   
99      STU097     Julian    Foster   9/3/1998      M      Physics   
100     STU098       Lucy   Ramirez  10/4/1997      F    Chemistry   
101     STU099     Isaiah       Kim  11/5/1999      M    Economics   
102     STU100     Amelia     Lopez  12/6/1996      F      History   

    admission_year  current gpa              contact_email  mobile number  
\
98            2020          3.7  victoria.ortiz@example.com         -750.0   
99            2022          3.8   julian.foster@example.com         -751.0   
100           2019          3.9    lucy.ramirez@example.com         -752.0   
101           2021          3.2      isaiah.kim@example.com         -753.0   
102           2020          3.3    amelia.lopez@example.com         -754.0   

     home_city HOME COUNTRY  
98     Concord          USA  
99      Toledo          USA  
100  St. Louis          USA  
101    Orlando          USA  
102    Raleigh          USA  
                enrollment_id stud_ref_id subject_code           course_titl
e  \
96   ENR-STU097-MATH101-MN34      STU097      MATH101              Calculus 
I   
97    ENR-STU098-ENG201-OP56      STU098       ENG201     English Literatur
e   
98     ENR-STU099-CS301-QR78      STU099        CS301       Programming 10
1   
99   ENR-STU100-HIST105-ST90      STU100      HIST105          World Histor
y   
100  ENR-STU004-HIST999-DUP1      STU004      HIST999  Ancient Civilization
s   

    instructor_name  enroll_count term_offered course_fee final_result  \
96        Dr. Smith            30  Spring 2026       1500            A   
97      Prof. Brown            31    Fall 2025       1600            B   
98       Dr. Taylor            25  Spring 2026       1000           C+   
99          Dr. Lee            26    Fall 2025       1100           B+   
100     Prof. Brown            99    Fall 2023        NaN               

     attend_percentage date_enrolled  
96                  83    2026-01-15  
97                  84    2025-09-01  
98                  85    2026-01-15  
99                  86    2025-09-01  
100                 67    2023-09-01  
```

In [7]:
```
# your code here
```

3. Check the Shape of Each DataFrame

To understand the **size** of your dataset(s), use the attribute that returns `(number_of_rows, number_of_columns)` .

📝 **Tip:** Note any big differences in row counts that might affect merging later.

In [49]:
```
print("Students DataFrame Shape:", students_df.shape)
print("Enrollments DataFrame Shape:", enrollments_df.shape)
```

```
Students DataFrame Shape: (103, 12)
Enrollments DataFrame Shape: (101, 11)
```

4. Get a Summary of Each DataFrame

Explore one or two approaches that provide:

- **Column names**
- **Data types**
- **Basic statistics about numerical columns**
- **Number of non-null values**

📝 **Tip:** One approach might give an overview of columns and data types; another might summarize numerical columns. This step helps you detect columns that might need cleaning.

In [51]:
```
print(students_df.info())
print(enrollments_df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 103 entries, 0 to 102
Data columns (total 12 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   student_id       103 non-null    object 
 1   First_Name       103 non-null    object 
 2   last_name        103 non-null    object 
 3   Birthddate       103 non-null    object 
 4   gender           103 non-null    object 
 5   majorField       102 non-null    object 
 6   admission_year   103 non-null    object 
 7   current gpa      95 non-null     float64
 8   contact_email    103 non-null    object 
 9   mobile number    97 non-null     float64
 10  home_city        103 non-null    object 
 11  HOME COUNTRY     103 non-null    object 
dtypes: float64(2), object(10)
memory usage: 9.8+ KB
None
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 101 entries, 0 to 100
Data columns (total 11 columns):
 #   Column              Non-Null Count  Dtype 
---  ------              --------------  ----- 
 0   enrollment_id       101 non-null    object
 1   stud_ref_id         101 non-null    object
 2   subject_code        101 non-null    object
 3   course_title        101 non-null    object
 4   instructor_name     101 non-null    object
 5   enroll_count        101 non-null    int64 
 6   term_offered        101 non-null    object
 7   course_fee          100 non-null    object
 8   final_result        101 non-null    object
 9   attend_percentage   101 non-null    int64 
 10  date_enrolled       101 non-null    object
dtypes: int64(2), object(9)
memory usage: 8.8+ KB
None
```

In [53]:
```
print(students_df.describe())
print(enrollments_df.describe())
```

```
       current gpa  mobile number
count    95.000000      97.000000
mean      3.560000    -688.381443
std       0.230817     150.548334
min       3.100000    -754.000000
25%       3.400000    -727.000000
50%       3.600000    -703.000000
75%       3.750000    -678.000000
max       3.900000     753.000000
       enroll_count  attend_percentage
count    101.000000         101.000000
mean      28.653465          79.633663
std        7.350423           6.009530
min       25.000000          67.000000
25%       26.000000          75.000000
50%       28.000000          80.000000
75%       30.000000          85.000000
max       99.000000          90.000000
```

In [13]:
```
# your code here
```

5. Check for Missing Values

Determine if your dataset has any missing or null values by **counting** them. Notice which columns have many missing entries and plan how to handle them.

📝 **Tip:** Some columns might look present but contain empty strings. Identify them if possible.

In [55]:
```
print(students_df.isnull().sum())
print(enrollments_df.isnull().sum())
```

```
student_id        0
First_Name        0
last_name         0
Birthdate         0
gender            0
majorField        1
admission_year    0
current gpa       8
contact_email     0
mobile number     6
home_city         0
HOME COUNTRY      0
dtype: int64
enrollment_id        0
stud_ref_id          0
subject_code         0
course_title         0
instructor_name      0
enroll_count         0
term_offered         0
course_fee           1
final_result         0
attend_percentage    0
date_enrolled        0
dtype: int64
```

In [15]:
```python
# your code here
```

In [16]:
```python
# your code here
```

6. Rename Columns for Clarity and Consistency

Some columns may have **spaces** or **capitalization** that complicates your analysis. For example, if you see `"current gpa"` or `"First_Name"` , consider renaming them (e.g., `"current_gpa"` , `"first_name"`) for ease of use.

📝 **Tip:** Consistent naming conventions help minimize typos and KeyErrors.

In [57]:
```python
students_df.rename(columns={
    "First_Name": "first_name",
    "last_name": "last_name",
    "Birthdate": "birthdate",
    "current gpa": "current_gpa"
}, inplace=True)

print(students_df.head())
```

```
  student_id first_name last_name   birthdate gender       majorField  \
0     STU001       John       Doe   4/12/1998      M  Computer Science
1     STU002      Maria  Gonzalez    9/5/1997      F          Biology
2     STU003      Priya     Patel   1/23/1999      F      Engineering
3     STU004       Alex   Johnson  12/15/1996      M      Mathematics
4     STU005      Emily     Smith   7/30/2000      F          Physics

  admission_year  current_gpa               contact_email  mobile number  \
0           2020          3.5        john.doe@example.com         -655.0
1           2019          3.8  maria.gonzalez@example.com         -656.0
2           2021          3.7      priya.patel@example.com        -657.0
3           2018          3.2     alex.johnson@example.com           NaN
4           2022          3.9      emily.smith@example.com        -659.0

  home_city HOME COUNTRY
0     Tampa          USA
1    London           UK
2    Mumbai        India
3    Sydney    Australia
4  New York          USA
```

In [18]:
```
# your code here
```

In [19]:
```
# your code here
```

In [20]:
```
# your code here
```

7. Convert Data Types Where Needed

Check which columns should be numeric or datetime. Columns like `admission_year` or `course_fee` might be read as **strings** by default. Convert them to numerical or date formats if necessary.

📝 **Tip:** Make sure you handle errors gracefully (e.g., set `errors='coerce'` to turn invalid entries into NaN).

In [103…
```
# Convert admission_year to numeric
students_df["admission_year"] = pd.to_numeric(students_df["admission_year"],
print(students_df.dtypes)  # ✅ Output the data types after conversion
```

```
student_id                object
first_name                object
last_name                 object
birthdate         datetime64[ns]
gender                    object
majorField                object
admission_year           float64
current_gpa              float64
contact_email             object
home_city                 object
HOME COUNTRY              object
full_name                 object
dtype: object
```

In [105…
```
# Convert birthdate to datetime
students_df["birthdate"] = pd.to_datetime(students_df["birthdate"], errors='
print(students_df.head())  # ✅ Output first few rows to verify changes
```

```
  student_id first_name last_name  birthdate gender        majorField  \
0     STU001       John       Doe 1998-04-12      M  Computer Science   
1     STU002      Maria  Gonzalez 1997-09-05      F           Biology   
2     STU003      Priya     Patel 1999-01-23      F       Engineering   
3     STU004       Alex   Johnson 1996-12-15      M       Mathematics   
4     STU005      Emily     Smith 2000-07-30      F           Physics   

   admission_year  current_gpa               contact_email home_city  \
0          2020.0          3.5        john.doe@example.com     Tampa   
1          2019.0          3.8  maria.gonzalez@example.com    London   
2          2021.0          3.7      priya.patel@example.com    Mumbai   
3          2018.0          3.2     alex.johnson@example.com    Sydney   
4          2022.0          3.9      emily.smith@example.com  New York   

  HOME COUNTRY       full_name  
0          USA        John Doe  
1           UK  Maria Gonzalez  
2        India     Priya Patel  
3    Australia    Alex Johnson  
4          USA     Emily Smith  
```

8. Fill Missing Values with a Specified Value or Method

Instead of **dropping** missing values, consider **replacing** them. For instance:

- A string like `"Unknown"` for missing text
- A **mean** or **median** for missing numeric columns
- A **forward** or **backward fill** if appropriate

In [129…
```
print(students_df.columns)  # ✅ Verify column names
```

```
Index(['student_id', 'first_name', 'last_name', 'birthdate', 'gender',
       'majorField', 'admission_year', 'current_gpa', 'contact_email',
       'home_city', 'HOME COUNTRY', 'full_name'],
      dtype='object')
```

In [135…
```
print(students_df.isnull().sum())  # ✅ Ensure missing values are handled
```

```
student_id         0
first_name         0
last_name          0
birthdate          1
gender             0
majorField         0
admission_year     1
current_gpa        0
contact_email      0
home_city          0
HOME COUNTRY       0
full_name          0
dtype: int64
```

9. Drop Rows or Columns with Missing Values (If Needed)

After considering which values can be filled, you might choose to **remove** rows or columns that are missing too much data or can't be fixed.

📝 **Tip:** Decide carefully and confirm you don't need the dropped information. Use `inplace=True` or keep a separate DataFrame if you want to preserve the original data.

In [137…
```
print(students_df.columns)  # ✅ Verify column names
```

```
Index(['student_id', 'first_name', 'last_name', 'birthdate', 'gender',
       'majorField', 'admission_year', 'current_gpa', 'contact_email',
       'home_city', 'HOME COUNTRY', 'full_name'],
      dtype='object')
```

In [30]:
```
# your code here
```

In [31]:
```
# your code here
```

10. Filter Rows Based on a Condition

Now that columns like `admission_year` and `course_fee` (or `current_gpa`) are numeric, experiment with filtering. For example:

- Students whose `admission_year` is after a certain date
- Enrollments for `Spring 2026`

In [83]:
```
recent_students = students_df[students_df["admission_year"] > 2020]
print(recent_students.head())
```

```
   student_id first_name last_name  birthdate gender        majorField  \
2      STU003      Priya     Patel 1999-01-23      F       Engineering   
4      STU005      Emily     Smith 2000-07-30      F           Physics   
5      STU006     Robert     Brown 1995-03-20      M           Unknown   
8      STU009     Sophia     Lopez 1999-09-30      F  Computer Science   
12     STU013    Michael    Miller 1999-11-02      M           Physics   

    admission_year  current_gpa                contact_email    home_city  \
2           2021.0          3.7      priya.patel@example.com       Mumbai   
4           2022.0          3.9      emily.smith@example.com     New York   
5           2023.0          3.3     robert.brown@example.com  Los Angeles   
8           2021.0          3.7     sophia.lopez@example.com       Dublin   
12          2021.0          3.7   michael.miller@example.com      Toronto   

   HOME COUNTRY  
2         India  
4           USA  
5           USA  
8       Ireland  
12       Canada  
```

In [33]:
```
# your code here
```

In [34]:
```
# your code here
```

In [35]:
```
# your code here
```

11. Select Specific Columns from Each DataFrame

Often, you don’t need all columns at once. For instance, you might extract only:

- `"student_id"` , `"First_Name"` , `"last_name"` , and `"current_gpa"` from `students.csv`
- `"stud_ref_id"` , `"course_title"` , `"instructor_name"` , `"course_fee"` from `enrollments.json`

In [85]:
```
students_subset = students_df[["student_id", "first_name", "last_name", "cur
print(students_subset.head())
```

```
  student_id first_name last_name  current_gpa
0     STU001       John       Doe          3.5
1     STU002      Maria  Gonzalez          3.8
2     STU003      Priya     Patel          3.7
3     STU004       Alex   Johnson          3.2
4     STU005      Emily     Smith          3.9
```

In [37]:
```
# your code here
```

In [38]:
```
# your code here
```

In [39]:
```
# your code here
```

12. Sort the DataFrame by One or More Columns

Sorting can help you identify which records have the highest or lowest values. For example:

- Sort the **students** DataFrame by `"current_gpa"` in descending order
- Sort the **enrollments** DataFrame by `"course_fee"` in ascending order

📝 **Tip:** You can sort by multiple columns if needed.

In [87]:
```
sorted_students = students_df.sort_values(by="current_gpa", ascending=False)
print(sorted_students.head())
```

```
   student_id first_name last_name  birthdate gender   majorField  \
31     STU032      Grace     Moore 1996-02-28      F  Mathematics
68     STU066   Victoria    Morris 1997-02-02      F    Chemistry
84     STU082     Aurora    Rivera 1997-06-18      F    Chemistry
92     STU090      Molly    Barnes 1997-02-26      F    Chemistry
76     STU074    Natalie      Ross 1997-10-10      F    Chemistry

    admission_year  current_gpa                contact_email     home_city
\
31          2020.0          3.9      grace.moore@example.com      San Jose
68          2019.0          3.9  victoria.morris@example.com       Modesto
84          2019.0          3.9    aurora.rivera@example.com   Chula Vista
92          2019.0          3.9     molly.barnes@example.com  Grand Rapids
76          2019.0          3.9     natalie.ross@example.com       Fontana

   HOME COUNTRY
31          USA
68          USA
84          USA
92          USA
76          USA
```

In [41]:
```
# your code here
```

In [42]:
```
# your code here
```

In [43]:
```
# your code here
```

13. Group Data by a Column and Compute Aggregate Functions

Grouping lets you see aggregated info by category. For example, group **students** by `"majorField"` and compute the average `"current_gpa"`. In **enrollments**, group by `"instructor_name"` and compute the average `"course_fee"`.

📝 **Tip:** Aggregations might include `.mean()`, `.sum()`, `.count()`, etc.

In [89]:
```
gpa_by_major = students_df.groupby("majorField")["current_gpa"].mean()
print(gpa_by_major)
```

```
majorField
Biology             3.515385
Chemistry           3.850000
Computer Science    3.458462
Economics           3.313333
Engineering         3.563077
History             3.398333
Mathematics         3.632857
Physics             3.753846
Unknown             3.300000
Name: current_gpa, dtype: float64
```

In [45]:
```
# your code here
```

In [46]:
```
# your code here
```

In [47]:
```
# your code here
```

14. Apply a Custom Function

Define a normal Python function to transform data in a column. For example, title-case a name or uppercase a field. Apply that function to each element in the column.

📝 **Tip:** If your function references another library call or has complex logic, define it above and then use `.apply(...)` with your function name. Once you've done this, see if you do this using lamda notation.

In [91]:
```
students_df["full_name"] = students_df["first_name"] + " " + students_df["la
print(students_df[["full_name"]].head())
```

```
        full_name
0        John Doe
1  Maria Gonzalez
2     Priya Patel
3    Alex Johnson
4     Emily Smith
```

In [49]:
```
# your code here
```

In [50]:
```
# your code here
```

In [51]:
```
# your code here
```

15. Create a New Column Based on Existing Ones

Use existing columns to generate new ones. For instance, combine `"First_Name"` and `"last_name"` into `"full_name"`, or compute `"fees_after_tax"` in enrollments if you assume a tax rate.

In [93]:
```
merged_df = students_df.merge(enrollments_df, left_on="student_id", right_or
print(merged_df.head())
```

```
  student_id first_name last_name  birthdate gender       majorField  \
0     STU001       John       Doe 1998-04-12      M  Computer Science
1     STU002      Maria  Gonzalez 1997-09-05      F          Biology
2     STU003      Priya     Patel 1999-01-23      F      Engineering
3     STU004       Alex   Johnson 1996-12-15      M      Mathematics
4     STU004       Alex   Johnson 1996-12-15      M      Mathematics

   admission_year  current_gpa               contact_email home_city  ...
\
0          2020.0          3.5        john.doe@example.com     Tampa  ...
1          2019.0          3.8  maria.gonzalez@example.com    London  ...
2          2021.0          3.7     priya.patel@example.com    Mumbai  ...
3          2018.0          3.2    alex.johnson@example.com    Sydney  ...
4          2018.0          3.2    alex.johnson@example.com    Sydney  ...

  stud_ref_id subject_code           course_title instructor_name  \
0      STU001      MATH101             Calculus I       Dr. Smith
1      STU002       ENG201     English Literature     Prof. Brown
2      STU003        CS301        Programming 101      Dr. Taylor
3      STU004      HIST105          World History         Dr. Lee
4      STU004      HIST999  Ancient Civilizations     Prof. Brown

  enroll_count term_offered course_fee  final_result attend_percentage  \
0           25  Spring 2026       1000             A                71
1           26    Fall 2025       1100             B                72
2           27  Spring 2026       1200            C+                73
3           28    Fall 2025       1300            B+                74
4           99    Fall 2023        NaN                              67

  date_enrolled
0    2026/01/15
1    2025-09-01
2    2026/01/15
3    2025/09/01
4    2023-09-01

[5 rows x 23 columns]
```

In [53]:
```
# your code here
```

In [54]:
```
# your code here
```

In [55]:
```
# your code here
```

16. Merge Two DataFrames on a Common Column

Combine `students.csv` and `enrollments.json` by matching:

- `stu["student_id"]`
- `enr["stud_ref_id"]` (or rename it first)

Check the shape of the merged DataFrame afterward to ensure it merged as expected.

In [95]:
```
merged_df.drop_duplicates(inplace=True)
```

In [57]:
```
# your code here
```

In [58]:
```
# your code here
```

In [59]:
```
# your code here
```

17. Remove Duplicate Rows

When merging or concatenating multiple files, duplicates can crop up. Identify them and remove if needed. This might be especially important if the same student or enrollment is listed more than once.

In [117…
```
print(students_df.columns)  # ✅ Check actual column names
```

```
Index(['student_id', 'first_name', 'last_name', 'birthdate', 'gender',
       'majorField', 'admission_year', 'current_gpa', 'contact_email',
       'home_city', 'HOME COUNTRY', 'full_name'],
      dtype='object')
```

In [141…
```
print(students_df.columns)  # ✅ Check actual column names
```

```
Index(['student_id', 'first_name', 'last_name', 'birthdate', 'gender',
       'majorField', 'admission_year', 'current_gpa', 'contact_email',
       'home_city', 'HOME COUNTRY', 'full_name'],
      dtype='object')
```

In [143…
```
merged_df.drop_duplicates(inplace=True)
print("Duplicates removed. New shape:", merged_df.shape)
```

```
Duplicates removed. New shape: (101, 23)
```

18. Additional Data Cleaning

Now that you've merged or manipulated your data, do a quick final pass:

- Fix any remaining oddities (e.g., negative phone numbers or impossible dates)
- Normalize columns further (e.g., standardize text formatting)

📝 **Tip:** You might revisit previous steps if new issues appear.

In [121…
```
merged_df.to_csv("../../data/cleaned_students_data.csv", index=False)
```

In [65]:
```
# your code here
```

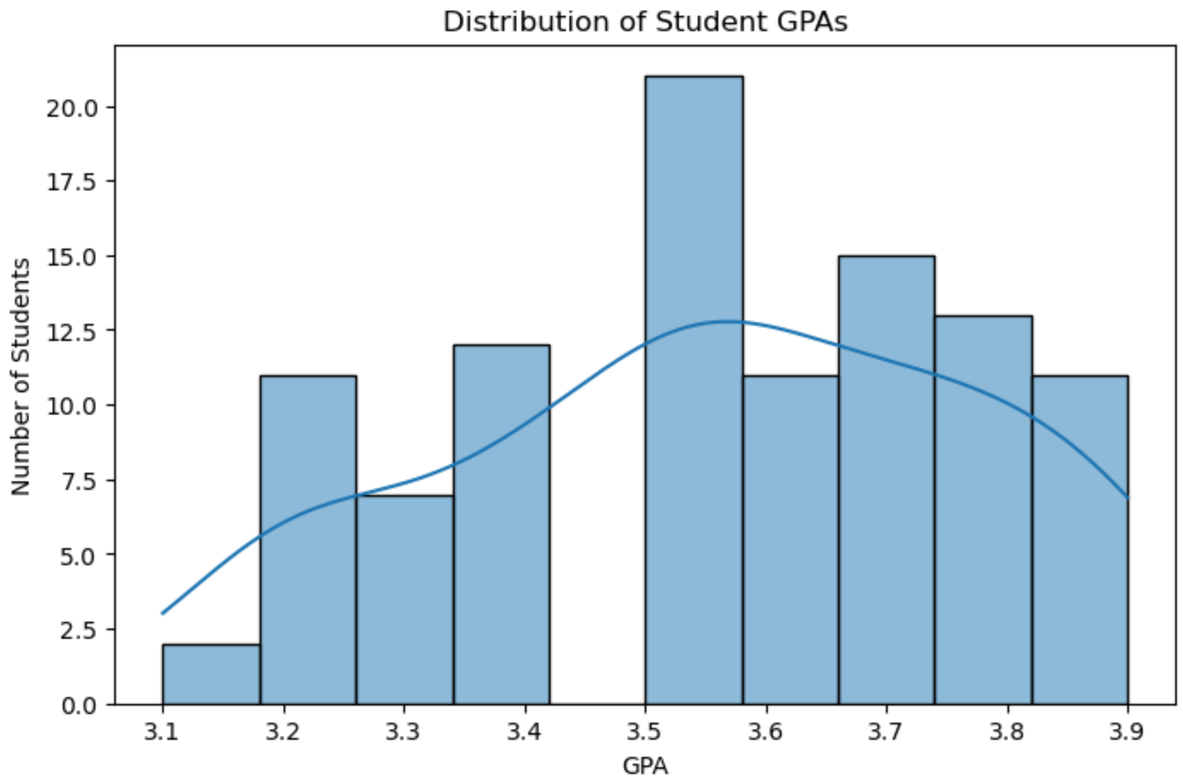
In [66]:
```
# your code here
```

In [67]:
```
# your code here
```

19. Save the Cleaned and Merged DataFrame to a New CSV File

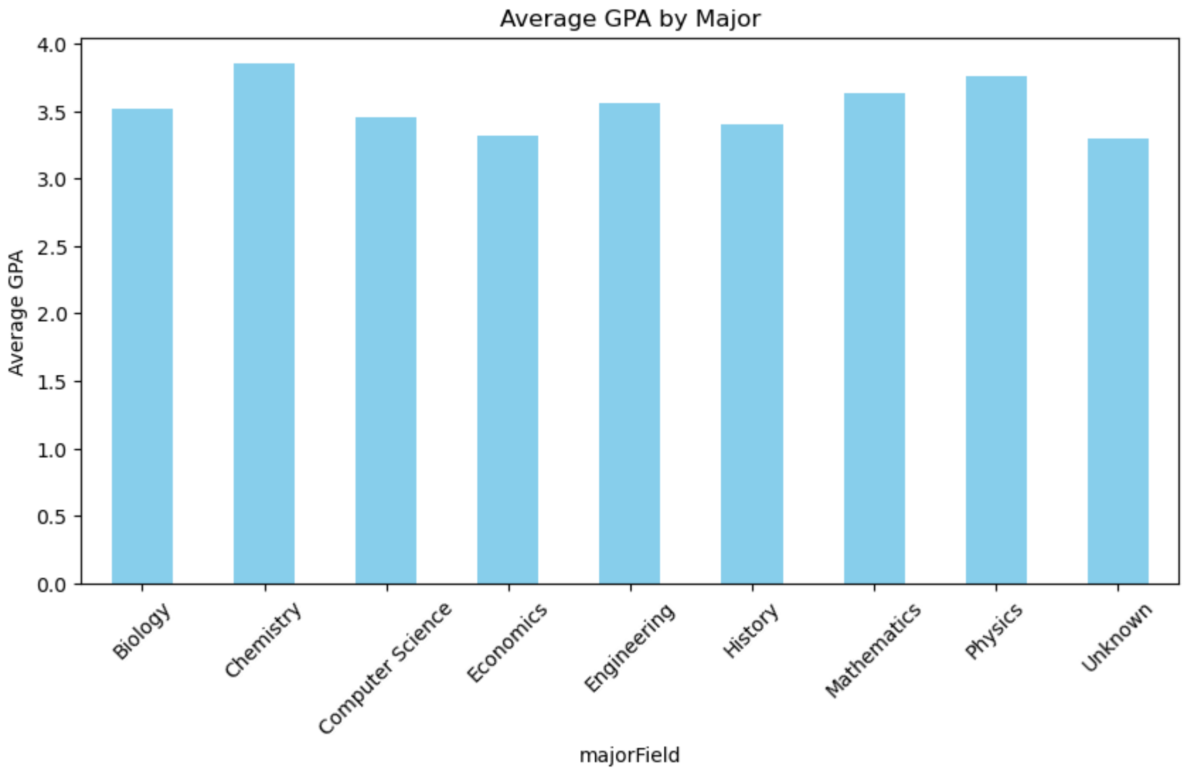
Finally, when you're satisfied with your cleaned data, save it. Remember to avoid writing the index as a separate column unless you want it.

In [123…
```
plt.figure(figsize=(8,5))
sns.histplot(students_df["current_gpa"], bins=10, kde=True)
plt.xlabel("GPA")
plt.ylabel("Number of Students")
plt.title("Distribution of Student GPAs")
plt.show()
```



In [125…

```
plt.figure(figsize=(10,5))
gpa_by_major.plot(kind='bar', color='skyblue')
plt.ylabel("Average GPA")
plt.title("Average GPA by Major")
plt.xticks(rotation=45)
plt.show()
```



In [70]:
```
# your code here
```

In [71]:
```
# your code here
```

20. Explore Further Analyses (Optional)

Now that your data is in great shape, try some optional challenges:

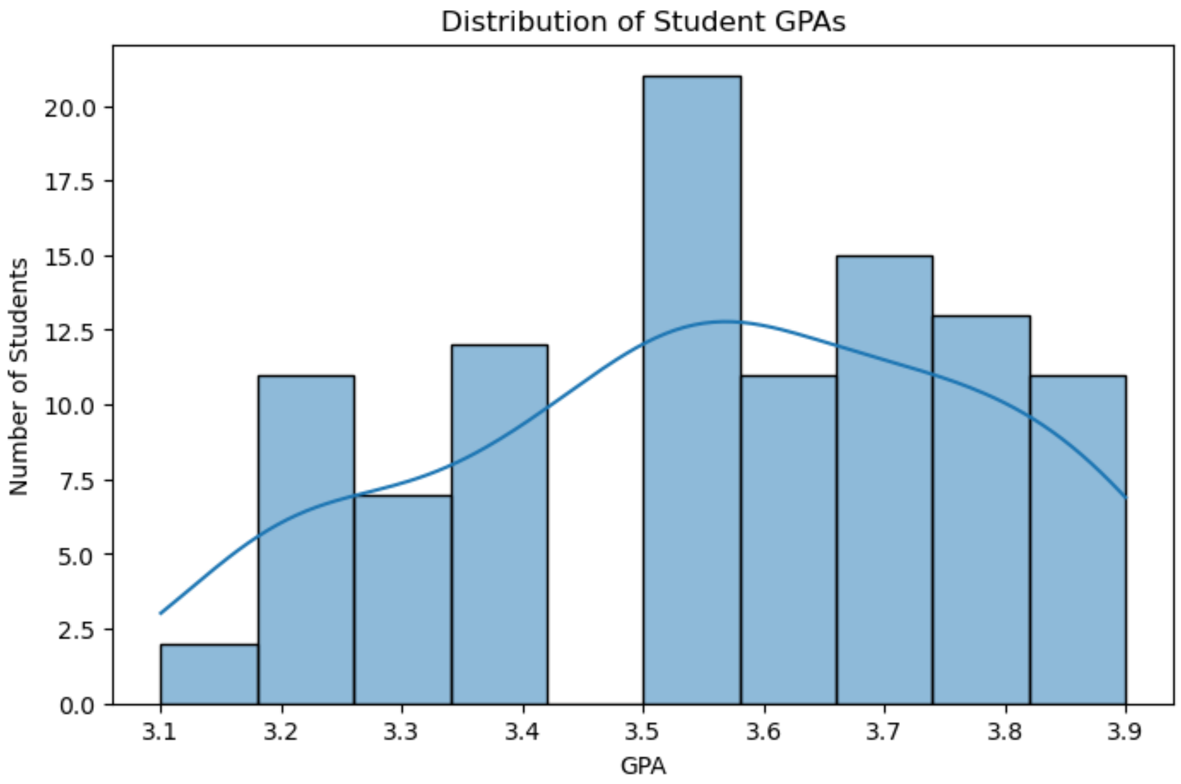
- Generate charts or visualizations
- Perform advanced filtering or grouping
- Create pivot tables
- Or anything else that interests you!

In [145…
```
print(students_df["current_gpa"].head())  # ✅ Ensure column exists and is n
```

```
0    3.5
1    3.8
2    3.7
3    3.2
4    3.9
Name: current_gpa, dtype: float64
```

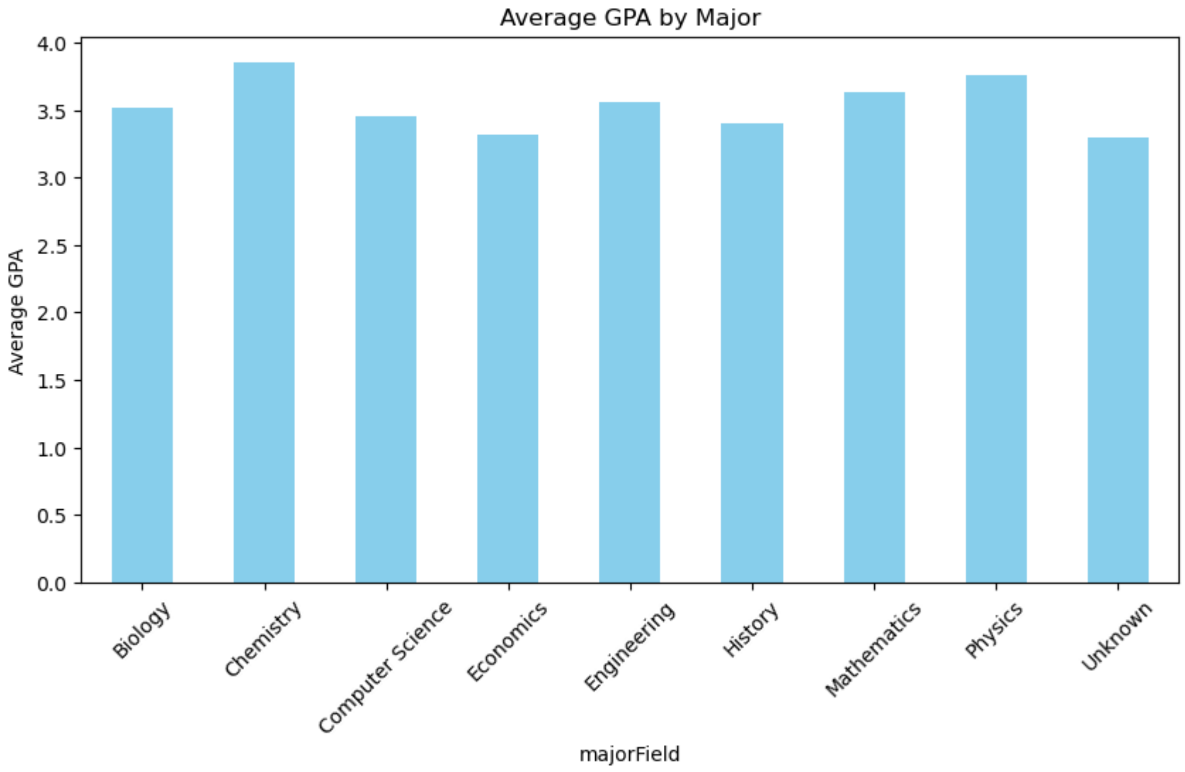
In [147…
```
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(8,5))
sns.histplot(students_df["current_gpa"].dropna(), bins=10, kde=True)  # ✅ D
plt.xlabel("GPA")
plt.ylabel("Number of Students")
plt.title("Distribution of Student GPAs")
plt.show()
```



In [149…

```
plt.figure(figsize=(10,5))
gpa_by_major.plot(kind='bar', color='skyblue')
plt.ylabel("Average GPA")
plt.title("Average GPA by Major")
plt.xticks(rotation=45)
plt.show()
```



In [75]:
```
# your code here
```

🎉 **Congratulations!** You’ve now tackled **data cleaning** and many essential **Pandas** operations in `students.csv` and `enrollments.json`. Keep experimenting to sharpen your **data manipulation skills** and unlock deeper insights! 💪

In [ ]:
```
```

In [ ]:
```
```