Data Manipulation with Pandas

SAURAB Gyawali
GitHub Instagram LinkedIn

November 29, 2025

Contents

Chapter 1

Data Manipulation with Pandas

1.1 Introduction Pandas

1.1.1 What is Pandas?

Pandas is a powerful Python library for data manipulation and analysis. It is designed to handle structured data, making it simple to perform cleaning, transformation, and analysis tasks.

Key Features

- Handles data from CSV, Excel, SQL, JSON and more.
- Offers Series (1D) and DataFrame (2D) structures.
- Provides filtering, aggregation, grouping, merging, and visualization tools.
- Seamlessly integrates with NumPy, Matplotlib, and Scikit-learn.

1.1.2 Creating a Series

A Series is a 1-dimensional labeled array capable of holding any data type.

```
import pandas as pd

# Series from list
s = pd.Series([10, 20, 30, 40], index=['A','B','C','D'])
print(s)

# Series from dictionary
s_dict = pd.Series({'Math':90,'Science':85,'English':88})
print(s_dict)
```

Output

```
A 10 B 20 C 30 D 40 dtype: int64
Math 90 Science 85 English 88 dtype: int64
```

1.1.3 Creating a DataFrame

A DataFrame is a 2-dimensional labeled data structure. It is like a spreadsheet in Python.

```python
# DataFrame from dictionary
data = {
    'Name':['Alice','Bob','Charlie'],
    'Age':[25,30,35],
    'Salary':[50000,60000,70000]
}
df = pd.DataFrame(data)
print(df)

# DataFrame from list of lists
data2 = [
    ['David', 28, 52000],
    ['Eva', 32, 65000]
]
df2 = pd.DataFrame(data2, columns=['Name','Age','Salary'])
print(df2)
```

Output

```
Name Age Salary 0 Alice 25 50000 1 Bob 30 60000 2 Charlie 35 70000
Name Age Salary 0 David 28 52000 1 Eva 32 65000
```

1.1.4 Accessing Data in DataFrame

```python
# Access column
print(df['Name'])

# Access multiple columns
print(df[['Name','Salary']])

# Access row by index
print(df.iloc[1])

# Access row by label
print(df.loc[0])
```

1.2 Data Import/Export (CSV, Excel, JSON)

1.2.1 Importing Data

```
# CSV
df_csv = pd.read_csv("data.csv")

# Excel
df_excel = pd.read_excel("data.xlsx", sheet_name=’Sheet1’)

# JSON
df_json = pd.read_json("data.json")
```

1.2.2 Exporting Data

```
# Export to CSV
df.to_csv("output.csv", index=False)

# Export to Excel
df.to_excel("output.xlsx", index=False)

# Export to JSON
df.to_json("output.json")
```

1.2.3 Example Dataset: Employees

Name	Age	Salary	Dept	JoinDate
Alice	25	50000	HR	2021-01-01
Bob	30	60000	IT	2020-05-12
Charlie	35	70000	Finance	2019-09-23
David	28	52000	IT	2022-03-15
Eva	32	65000	HR	2021-07-20

1.3 Data Cleaning: Handling Missing Values, Filtering, Sorting

1.3.1 Detect Missing Values

```
# Check for missing values
df.isnull().sum()
```

1.3.2 Filling or Dropping Missing Values

```
# Fill missing Age with mean
df['Age'].fillna(df['Age'].mean(), inplace=True)

# Drop rows with missing Salary
df.dropna(subset=['Salary'], inplace=True)
```

1.3.3 Filtering Data

```
# Employees in IT department
it_dept = df[df['Dept'] == 'IT']

# Employees with Salary > 60000
high_salary = df[df['Salary'] > 60000]

# Employees in HR or IT
dept_hr_it = df[df['Dept'].isin(['HR','IT'])]
```

1.3.4 Sorting Data

```
# Sort by Age ascending
df_sorted = df.sort_values(by='Age')

# Sort by Salary descending
df_sorted_salary = df.sort_values(by='Salary', ascending=False)

# Sort by Dept and then Salary
df_sorted_multi = df.sort_values(by=['Dept','Salary'], ascending=[
   True, False])
```

1.4 Data Transformation: Grouping, Merging, Joining, and Aggregation

1.4.1 Grouping and Aggregation

```
# Average salary by Department
dept_salary = df.groupby('Dept')['Salary'].mean()

# Count of employees per department
dept_count = df.groupby('Dept')['Name'].count()

# Multiple aggregations
```

```
dept_stats = df.groupby('Dept').agg({
    'Salary':'mean',
    'Age':'max'
})
```

1.4.2 Merging and Joining

```
df1 = pd.DataFrame({'ID':[1,2,3],'Name':['A','B','C']})
df2 = pd.DataFrame({'ID':[2,3,4],'Salary':[50000,60000,70000]})

# Outer merge
merged_df = pd.merge(df1, df2, on='ID', how='outer')

# Left merge
left_merge = pd.merge(df1, df2, on='ID', how='left')

# Right merge
right_merge = pd.merge(df1, df2, on='ID', how='right')
```

1.4.3 Adding Calculated Columns

```
# Bonus = 10% of Salary
df['Bonus'] = df['Salary'] * 0.10

# Total Compensation = Salary + Bonus
df['TotalComp'] = df['Salary'] + df['Bonus']

# Age group
df['AgeGroup'] = pd.cut(df['Age'], bins=[20,30,40], labels=['Young',
   'Adult'])
```

1.5 Descriptive Statistics and Data Summarization

1.5.1 Summary Statistics

```
df.describe()         # Summary of numerical columns
df.describe(include='all')   # Summary of all columns
```

1.5.2 Correlation and Covariance

```
df.corr()    # Correlation matrix
df.cov()     # Covariance matrix
```

1.5.3 Value Counts and Unique Values

```python
df['Dept'].value_counts()
df['Dept'].unique()
```

1.5.4 Visualization Examples

```python
import matplotlib.pyplot as plt

# Histogram of Age
df['Age'].plot(kind='hist', bins=5, title='Age Distribution')
plt.show()

# Scatter plot Salary vs Age
df.plot(kind='scatter', x='Age', y='Salary', title='Salary vs Age')
plt.show()

# Box plot for Salary by Department
df.boxplot(column='Salary', by='Dept')
plt.show()

# Bar plot for number of employees per Dept
df['Dept'].value_counts().plot(kind='bar', title='Employees per Department')
plt.show()
```