Pandas Notes

Introduction to Pandas:
Pandas is an open-source Python library that provides high-performance, easy-to-use data structures and data analysis tools. It's built on top of NumPy and is particularly suited for working with structured or tabular data.

To use Pandas, you typically import it like this:

```
import pandas as pd
```

Pandas is a powerful tool for data manipulation and analysis in Python. It provides easy-to-use data structures and functions to handle various data formats efficiently. By mastering Pandas, you can effectively work with structured data in Python for tasks like data cleaning, manipulation, and analysis.

Series Data Structure:
The Series is a one-dimensional labeled array capable of holding any data type (integers, strings, floats, Python objects, etc.). It's similar to a one-dimensional NumPy array, but with labels for each element called the index.
Here's how you can create a Series:

```
# Create a Series
data = pd.Series([1, 2, 3, 4, 5], index=['a', 'b', 'c', 'd', 'e'])
print(data)
```

Querying and Indexing Series:
You can access elements in a Series using their index labels:

```
# Accessing elements
print(data['a'])  # Output: 1
```

You can also perform boolean indexing:

```
# Boolean indexing
print(data[data > 2])  # Output: b    3  c    4  d    5  dtype: int64
```

DataFrame Data Structure:
The DataFrame is a two-dimensional labeled data structure with columns of potentially different types. It's like a spreadsheet or SQL table, or a dictionary of Series objects.
Here's how you can create a DataFrame:

```
import pandas as pd
# Create a DataFrame
data = {'Name': ['Alice', 'Bob', 'Charlie'],
        'Age': [25, 30, 35],
        'Salary': [50000, 60000, 70000]}
df = pd.DataFrame(data)
print(df)
```

Querying and Indexing DataFrame:
You can access columns by their names:

```
# Accessing columns
print(df['Name'])  # Output: 0      Alice  1        Bob  2    Charlie  Name: Name, dtype: object
```

You can also perform boolean indexing on DataFrames:

```
# Boolean indexing
print(df[df['Age'] > 30])  # Output:      Name  Age  Salary  2  Charlie  35  70000
```

Loading Data:
Pandas supports reading data from various file formats such as CSV, Excel, SQL databases, and more.

```
# Reading CSV file
df = pd.read_csv('data.csv')

# Reading Excel file
df = pd.read_excel('data.xlsx')

# Reading from SQL database
import sqlite3
conn = sqlite3.connect('database.db')
df = pd.read_sql_query("SELECT * FROM table_name", conn)
```

Merging DataFrames:
Merging DataFrames is a way to combine two or more DataFrames based on one or more keys. There are several types of joins you can perform, such as inner join, outer join, left join, and right join.

Here's an example:

```
import pandas as pd

# Create two DataFrames
df1 = pd.DataFrame({'A': ['A0', 'A1', 'A2', 'A3'],
                    'B': ['B0', 'B1', 'B2', 'B3'],
                    'key': ['K0', 'K1', 'K2', 'K3']})

df2 = pd.DataFrame({'C': ['C0', 'C1', 'C2', 'C3'],
                    'D': ['D0', 'D1', 'D2', 'D3'],
                    'key': ['K0', 'K1', 'K2', 'K3']})

# Merge DataFrames on 'key' column
merged_df = pd.merge(df1, df2, on='key')

print(merged_df)
```

Group By Operation:
The groupby operation involves splitting the data into groups based on some criteria, applying a function to each group independently, and then combining the results.

```
# Grouping by 'A' column and calculating mean of 'B' column
grouped = df.groupby('A').mean()

print(grouped)
```

Pivot Table:
Pivot tables are used to summarize and aggregate data inside a DataFrame. They provide a way to reshape and summarize data.

```
# Creating a pivot table
pivot_table = df.pivot_table(values='D', index='A', columns='B', aggfunc='count')
```

```
print(pivot_table)
```

Date/Time Functionality:
Pandas provides robust functionality for working with dates and times. You can create, manipulate, and convert date/time objects.

```
# Creating a DataFrame with dates
dates = pd.date_range('2022-01-01', periods=5)
df_dates = pd.DataFrame({'Date': dates, 'Value': [1, 2, 3, 4, 5]})

# Extracting year, month, day
df_dates['Year'] = df_dates['Date'].dt.year
df_dates['Month'] = df_dates['Date'].dt.month
df_dates['Day'] = df_dates['Date'].dt.day

print(df_dates)
```

Example: Manipulating DataFrame:

```
# Creating a DataFrame
data = {'Name': ['Alice', 'Bob', 'Charlie'],
        'Age': [25, 30, 35],
        'Salary': [50000, 60000, 70000]}

df = pd.DataFrame(data)

# Adding a new column
df['Department'] = ['HR', 'Finance', 'IT']

# Filtering rows based on condition
filtered_df = df[df['Age'] > 30]

print(filtered_df)
```