Class 3: Pandas DataFrame Creation and Analysis

Overview

Pandas is a Python library for working with tabular data, enabling analysis, cleaning, exploration, and manipulation. This class note covers the creation and analysis of Pandas DataFrames, including methods for data manipulation, grouping, and pivoting.

Objectives

- Understand Pandas DataFrame creation and basic methods (`head`, `tail`, `shape`, `info`, `describe`, `values`, `columns`).
- Learn sorting, selecting columns/rows, and creating new columns.
- Explore DataFrame queries, grouping, and pivot tables.
- Apply custom methods and analyze categorical variables.

Key Concepts

1. **DataFrame Creation**: Create DataFrames from dictionaries or files (e.g., `.csv`, `.json`).
2. **DataFrame Methods**:
   - `head()`: View first 5 rows (or specified number).
   - `tail(n)`: View last n rows.
   - `shape`: Returns dimensions (rows, columns).
   - `info()`: Summarizes DataFrame structure.
   - `describe()`: Provides statistical summary.
   - `values`: Converts DataFrame to NumPy array.
   - `columns`: Lists column names.
3. **Sorting and Selection**:
   - Sort by columns: `sort_values(by=["col"], ascending=True/False)`.
   - Select columns: `df["col"]` or `df[["col1", "col2"]]`.
   - Select rows: Use boolean masks (e.g., `df[df["col"] > value]`).
4. **New Columns**: Add derived columns (e.g., `df["total_pop"] = df["col1"] + df["col2"]`).
5. **Query**: Filter rows using `query("condition")` (e.g., `region == 'Pacific' and family_members < 10000`).
6. **Categorical Analysis**: Use `value_counts()` to count unique values.
7. **Grouping**: Group data with `groupby("col")`, apply aggregations like `sum()`, `mean()`.
8. **Pivot Tables**: Summarize data with `pivot_table(values, index, columns, aggfunc)`.

Example Code

```
import pandas as pd
import numpy as np

% Creating a DataFrame from a dictionary
health_data = {
    ”height”: [5.5, 5.5, 6.0, 5.8],
    ”weight”: [57, 62, 72, 75],
    ”bmi”: [19.3, 12.7, 20.3, 18.5]
}
df = pd.DataFrame(health_data)

% Basic DataFrame methods
print(df.head(2))  % Display first 2 rows
print(df.tail(1))  % Display last row
print(df.shape)    % Output: (4, 3)
print(df.info())   % Display DataFrame info

% Sorting by multiple columns
sorted_df = df.sort_values(by=[”height”, ”weight”], ascending=[True, False])

% Filtering rows with a condition
filtered_df = df[df[”bmi”] > 19]

% Creating a new column
df[”height_meters”] = df[”height”] * 0.3048  % Convert feet to meters

% Reading a CSV file
sales = pd.read_csv(”sales_subset.csv”)

% Dropping unnecessary columns
sales = sales.drop(columns=[”Unnamed: 0”])

% Statistical analysis
print(sales[”weekly_sales”].mean())  % Average weekly sales
print(sales[”weekly_sales”].median())  % Median weekly sales

% Applying custom function to normalize data
def normalize(col):
    mean = col.mean()
    std = col.std()
    return (col - mean) / std

normalized_sales = sales[[”weekly_sales”, ”temperature_c”]].agg([normalize, np.
    cumsum])

% Counting categorical variables
print(sales[”type”].value_counts())  % Count unique store types

% Grouping by store type
sales_grouped = sales.groupby(”type”)[”weekly_sales”].agg([min, max, np.mean, np.
    median])

% Creating a pivot table
pivot = sales.pivot_table(
    values=”weekly_sales”,
    index=”type”,
    columns=”is_holiday”,
    aggfunc=np.mean
)
```

Key Takeaways

Pandas provides powerful tools for data manipulation and analysis. Mastering DataFrame creation, methods, sorting, grouping, and pivot tables enables efficient handling of tabular data for real-world applications.