**In data engineering**, certain **pandas methods** are particularly useful for handling large datasets, transforming data, ensuring data quality, and preparing data for analysis. Here are some key pandas methods that are frequently used in data engineering tasks:

**### Data Loading**

- \*\*`pd.read\_csv(filepath\_or\_buffer, ...)`\*\*: Load data from a CSV file.

- \*\*`pd.read\_excel(io, ...)`\*\*: Load data from an Excel file.

- \*\*`pd.read\_sql(sql, con, ...)`\*\*: Load data from a SQL database.

**### Data Inspection and Exploration**

- \*\*`df.head(n)`\*\*: Preview the first n rows of the DataFrame.

- \*\*`df.tail(n)`\*\*: Preview the last n rows of the DataFrame.

- \*\*`df.info()`\*\*: Get a concise summary of the DataFrame.

- \*\*`df.describe()`\*\*: Generate descriptive statistics.

- \*\*`df.shape`\*\*: Get the dimensions of the DataFrame.

- \*\*`df.columns`\*\*: Get the column labels.

- \*\*`df.dtypes`\*\*: Get the data types of each column.

**### Data Selection and Filtering**

- \*\*`df.loc[]`\*\*: Access a group of rows and columns by labels.

- \*\*`df.iloc[]`\*\*: Access a group of rows and columns by integer positions.

- \*\*`df.query(expr)`\*\*: Query the columns of the DataFrame with a boolean expression.

**### Data Cleaning and Preparation**

- \*\*`df.drop(labels, axis)`\*\*: Drop specified labels from rows or columns.

- \*\*`df.dropna(axis, how, ...)`\*\*: Remove missing values.

- \*\*`df.fillna(value, ...)`\*\*: Fill missing values.

- \*\*`df.replace(to\_replace, value, ...)`\*\*: Replace values.

- \*\*`df.astype(dtype, ...)`\*\*: Convert the DataFrame to a specific dtype.

- \*\*`df.duplicated(subset, keep)`\*\*: Identify duplicate rows.

- \*\*`df.drop\_duplicates(subset, keep)`\*\*: Remove duplicate rows.

- \*\*`df.rename(columns, index, ...)`\*\*: Rename labels.

- \*\*`df.set\_index(keys, ...)`\*\*: Set the DataFrame index using existing columns.

- \*\*`df.reset\_index(level, ...)`\*\*: Reset the index.

**### Data Transformation**

- \*\*`df.sort\_values(by, ...)`\*\*: Sort the DataFrame by the values along either axis.

- \*\*`df.sort\_index(axis, ...)`\*\*: Sort the DataFrame by index.

- \*\*`df.apply(func, axis, ...)`\*\*: Apply a function along an axis of the DataFrame.

- \*\*`df.applymap(func)`\*\*: Apply a function element-wise to the DataFrame.

- \*\*`df.transform(func, ...)`\*\*: Apply a function that returns a DataFrame with the same shape.

- \*\*`df.pivot(index, columns, values)`\*\*: Pivot the DataFrame (reshape data).

- \*\*`df.pivot\_table(values, index, columns, ...)`\*\*: Create a pivot table.

**### Data Aggregation and Grouping**

- \*\*`df.groupby(by, ...)`\*\*: Group DataFrame using a mapper or by a series of columns.

- \*\*`df.agg(func, axis, ...)`\*\*: Aggregate using one or more operations over the specified axis.

- \*\*`df.mean(axis, skipna)`\*\*: Compute mean.

- \*\*`df.sum(axis, skipna)`\*\*: Compute sum.

- \*\*`df.count(axis)`\*\*: Count non-NA cells.

- \*\*`df.min(axis, skipna)`\*\*: Compute minimum.

- \*\*`df.max(axis, skipna)`\*\*: Compute maximum.

**### Joining and Merging**

- \*\*`df.merge(right, ...)`\*\*: Merge DataFrame or named Series objects with a database-style join.

- \*\*`df.join(other, ...)`\*\*: Join columns of another DataFrame.

- \*\*`pd.concat(objs, ...)`\*\*: Concatenate pandas objects along a particular axis.

**### Reshaping and Pivoting**

- \*\*`df.melt(id\_vars, value\_vars, ...)`\*\*: Unpivot a DataFrame from wide format to long format.

- \*\*`df.stack()`\*\*: Stack the prescribed level(s) from columns to index.

- \*\*`df.unstack(level)`\*\*: Unstack, moving the index to columns.

**### Time Series and Window Operations**

- \*\*`df.resample(rule, ...)`\*\*: Resample time-series data.

- \*\*`df.rolling(window, ...)`\*\*: Provide rolling window calculations.

- \*\*`df.expanding(min\_periods, ...)`\*\*: Provide expanding transformations.

- \*\*`df.ewm(span, ...)`\*\*: Provide exponential weighted functions.

**### Input/Output Operations**

- \*\*`df.to\_csv(path\_or\_buf, ...)`\*\*: Write DataFrame to a CSV file.

- \*\*`df.to\_excel(excel\_writer, ...)`\*\*: Write DataFrame to an Excel file.

- \*\*`df.to\_json(path\_or\_buf, ...)`\*\*: Write DataFrame to a JSON string or file.

- \*\*`df.to\_sql(name, con, ...)`\*\*: Write records stored in a DataFrame to a SQL database.

**### Performance Optimization**

- \*\*`df.memory\_usage(index, ...)`\*\*: Return the memory usage of each column.

- \*\*`df.sample(n, ...)`\*\*: Return a random sample of items from an axis.

- \*\*`df.corr(method)`\*\*: Compute pairwise correlation of columns.

- \*\*`df.cov()`\*\*: Compute pairwise covariance of columns.

**### Example Workflow in Data Engineering**

Here's an example of a data engineering workflow using some of these methods:

import pandas as pd

# Load data

df = pd.read\_csv('data.csv')

# Inspect data

print(df.info())

print(df.describe())

# Data cleaning

df = df.dropna(subset=['important\_column']) # Remove rows with missing important values

df['category'] = df['category'].astype('category') # Optimize memory usage

# Data transformation

df['total'] = df['quantity'] \* df['price'] # Create a new column

# Data aggregation

agg\_df = df.groupby('category').agg({

'total': 'sum',

'quantity': 'mean'

}).reset\_index()

# Data export

agg\_df.to\_csv('aggregated\_data.csv', index=False)

```

These methods help ensure data is properly cleaned, transformed, and aggregated, making it ready for further analysis or use in downstream systems.