Python Project: Mini-Pandas

Objective

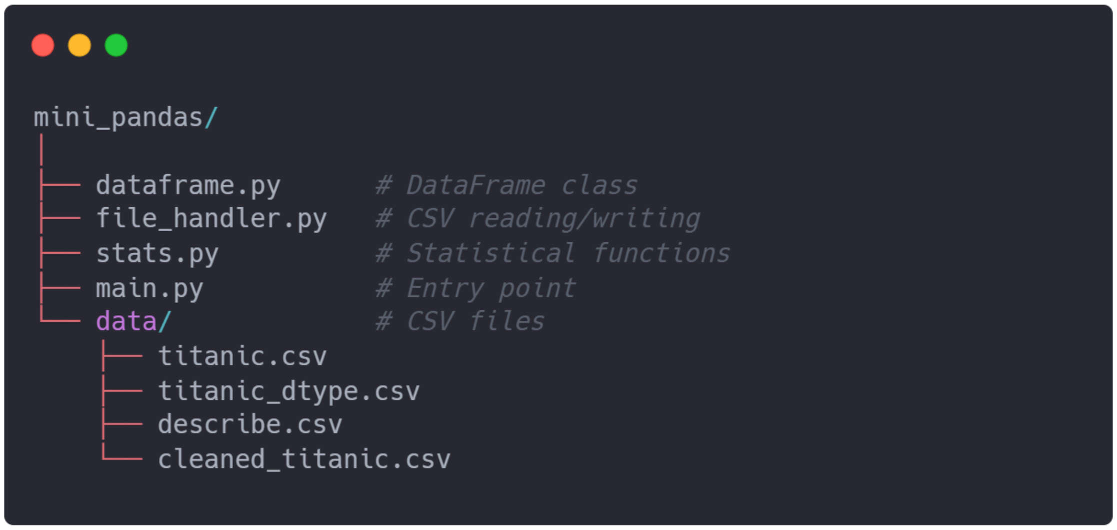
The goal of this project is to create a **mini version of the pandas DataFrame library** in Python.

Project Requirements

1. **No imports allowed except `functools.reduce, csv`, `your own defined modules.`**
2. The project should be modular (using separate files):
    - `dataframe.py` → DataFrame class
    - `file_handler.py` → CSV reading/writing
    - `stats.py` → Statistical functions

3. Code should be **object-oriented**. The main class is `Dataframe`.
4. Generate outputs as CSV files
5. Any function that is orginally meant to accept numerical values you should handle the case if I tried to pass a strings [exception handling]

Architecture



```
mini_pandas/
│
├── dataframe.py       # DataFrame class
├── file_handler.py    # CSV reading/writing
├── stats.py           # Statistical functions
├── main.py            # Entry point
└── data/              # CSV files
    ├── titanic.csv
    ├── titanic_dtype.csv
    ├── describe.csv
    └── cleaned_titanic.csv
```

File	**Responsibility**
`dataframe.py`	Contains `Dataframe` class, orchestrates operations on the dataset
`file_handler.py`	Functions to read/write CSV files
`stats.py`	Pure statistical functions (max, min, mean, median, mode)
`main.py`	Entry point for running the pipeline
`data/`	Stores input and output CSV files

Module: `file_handler.py`

Implement the following functions:

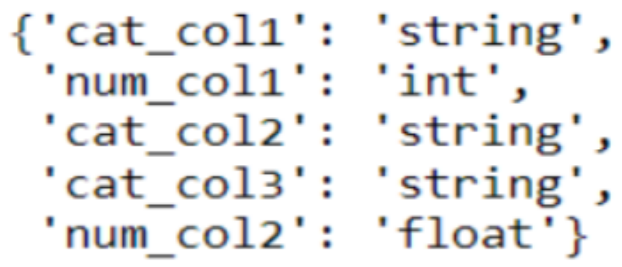
1️⃣ `read_dtype`

Purpose: Read a CSV file containing column names and their data types.
Args:

- `file_path` **(str): Path to the data types CSV file.**

Returns:

- `dict`**: Keys are column names, values are data types (**`int`**,** `float`**,** `string`**).**



```
{'cat_col1': 'string',
 'num_col1': 'int',
 'cat_col2': 'string',
 'cat_col3': 'string',
 'num_col2': 'float'}
```

2️⃣ `read_csv_file`

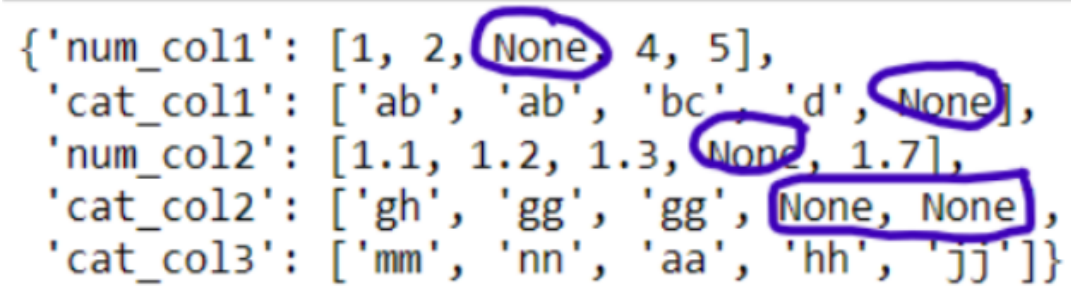
Purpose: Read a CSV file and convert each column to the specified data type.
Args:

- `file_path` (str): Path to the CSV data file.

- `dtypes` (dict): Dictionary mapping column names to data types.

Returns:

- `dict`: Keys are column names, values are lists of column values. Missing values are replaced with None.



```
{'num_col1': [1, 2, None, 4, 5],
 'cat_col1': ['ab', 'ab', 'bc', 'd', None],
 'num_col2': [1.1, 1.2, 1.3, None, 1.7],
 'cat_col2': ['gh', 'gg', 'gg', None, None],
 'cat_col3': ['mm', 'nn', 'aa', 'hh', 'jj']}
```

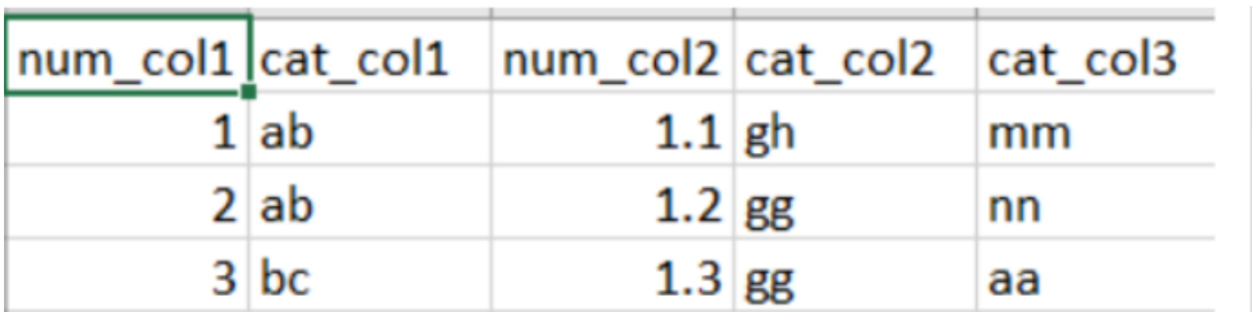
3️⃣`write_file`

Purpose: Write a data dictionary to a CSV file.
Args:

- `file_path` (str): Path to the output CSV file.
- `data` (dict): Dictionary where keys are column names and values are lists of column values.

Returns:

- None. Writes a CSV file to disk.
- Output should be in this format:



num_col1	cat_col1	num_col2	cat_col2	cat_col3
1	ab	1.1	gh	mm
2	ab	1.2	gg	nn
3	bc	1.3	gg	aa

Module: `stats.py`

Implement the following functions:

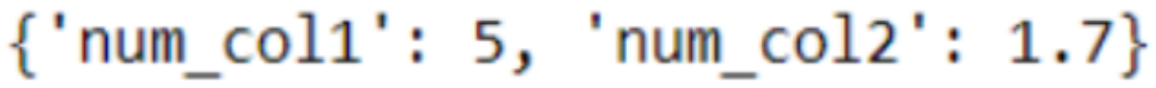
4️⃣`get_col_max`

Purpose: Return the maximum value of a numerical column.
Args:

- **`col` (list): List of numerical values. None values are ignored.**

Returns:

- **Numeric: Maximum value in the column.**
- **Note: don’t use any max or sort or sorted functions (but you can use filter,map or reduce if that helps).**



```
{'num_col1': 5, 'num_col2': 1.7}
```

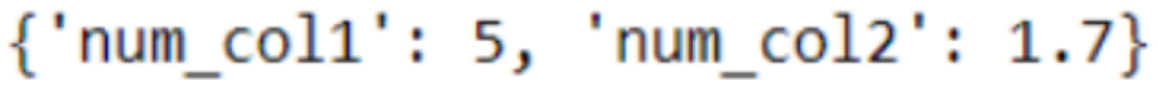
5️⃣`get_col_min`

Purpose: Return the minimum value of a numerical column.
Args:

- **`col` (list): List of numerical values. None values are ignored.**

Returns:

- **Numeric: Minimum value in the column.**
- **Note: don’t use any max or sort or sorted functions (but you can use filter,map or reduce if that helps).**



```
{'num_col1': 5, 'num_col2': 1.7}
```

6️⃣ `get_col_mean`

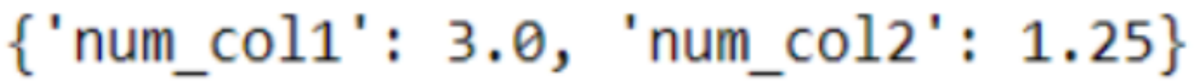
Purpose: Return the mean (average) value of a numerical column.

Args:

- **`col` (list): List of numerical values. `None` values are ignored.**

Returns:

- **Float: Mean value of the column.**
- **Note: don’t use any max or sort or sorted functions (but you can use filter,map or reduce if that helps).**



```
{'num_col1': 3.0, 'num_col2': 1.25}
```

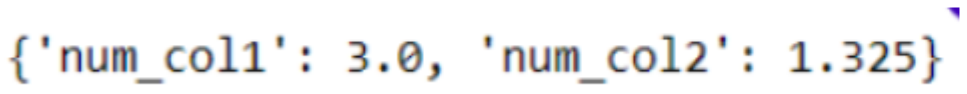
7️⃣ `get_col_median`

Purpose: Return the median value of a numerical column.
Args:

- `col` (list): List of numerical values. `None` values are ignored.

Returns:

- **Numeric: Median value of the column.**



```
{'num_col1': 3.0, 'num_col2': 1.325}
```

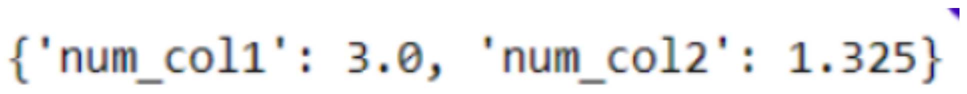
8️⃣`get_col_mode`

Purpose: Return the mode (most frequent value) of a column.
Args:

- `col` (list): List of values. `None` values are ignored.

Returns:

- **Value: The mode value of the column. Returns the first encountered if multiple modes exist.**



```
{'num_col1': 3.0, 'num_col2': 1.325}
```

9️⃣`get_stat`

Purpose: Apply a statistical function to all numerical columns in a dataset.
Args:

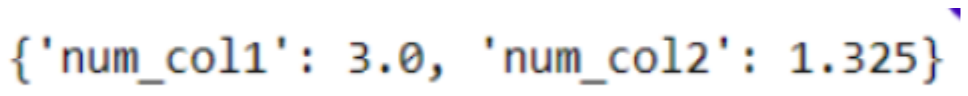
- `data` (dict): Dictionary where keys are column names and values are lists of column values.

- `dtypes` (dict): Dictionary where keys are column names and values are data types.

- `function` (str): Name of the statistical function to apply (e.g., `'get_col_max'`).

Returns:

- dict: Keys are column names, values are the result of applying the function. Only numerical columns are processed.



```
{'num_col1': 3.0, 'num_col2': 1.325}
```

Module: `dataframe.py`

Implement the following functions:

0️⃣`__init__`

Purpose: Initialize a DataFrame instance with data and data types.
Args:

- `data` (dict): Column name → list of values.
- `dtype` (dict): Column name → data type (`int`, `float`, `string`).

Returns:

- None.

1️⃣`read_csv` (classmethod)

Purpose: Read CSV data and dtype file and return a DataFrame instance.
Args:

- `data_path` (str): Path to the CSV data file.
- `dtype_path` (str): Path to the CSV dtype file.

Returns:

- `Dataframe`: A new DataFrame instance with data loaded.

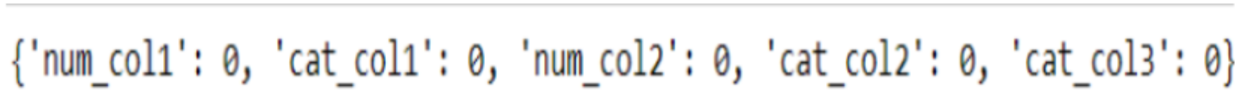
2️⃣`count_nulls`

Purpose: Count the number of missing values in each column.
Args:

- None.

Returns:

- **dict: Column name → number of missing (`None`) values.**



```
{'num_col1': 0, 'cat_col1': 0, 'num_col2': 0, 'cat_col2': 0, 'cat_col3': 0}
```

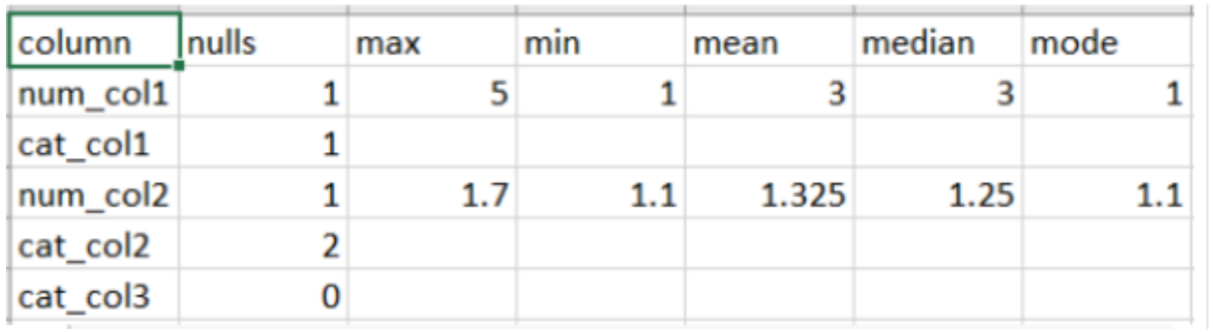
3️⃣ `describe`

Purpose: Generate a CSV file containing statistics for each column.
Args:

- `path` **(str, optional): Path to save the CSV. Default:** `'data/describe.csv'`**.**

Returns:

- **None. Writes CSV with columns:** `column`, `nulls`, `max`, `min`, `mean`, `median`, `mode`**.**
- **Output should be in this format:**



column	nulls	max	min	mean	median	mode
num_col1	1	5	1	3	3	1
cat_col1	1					
num_col2	1	1.7	1.1	1.325	1.25	1.1
cat_col2	2					
cat_col3	0					

4️⃣ `fillna`

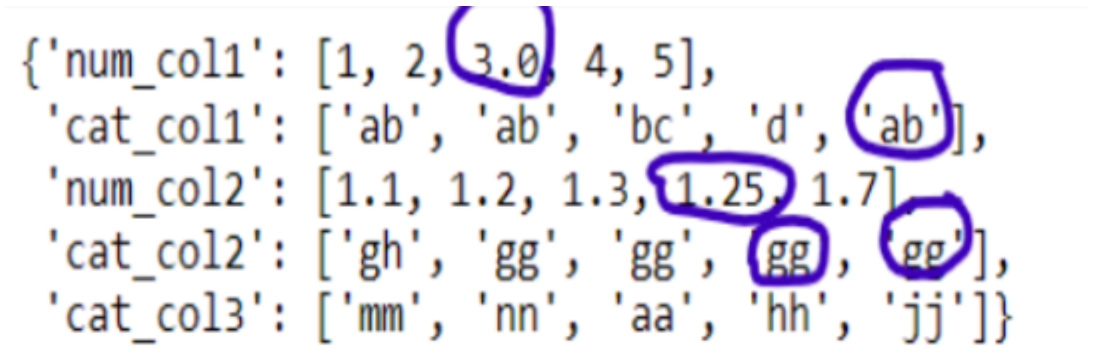
Purpose: Fill missing values in the DataFrame.

Args:

- `num_strategy` (function): Function to fill numeric columns (`get_col_mean`, `get_col_median`, etc.).
- `cat_strategy` (function): Function to fill categorical columns (`get_col_mode`).

Returns:

- None. Updates `self.data` in place.



```
{'num_col1': [1, 2, 3.0, 4, 5],
 'cat_col1': ['ab', 'ab', 'bc', 'd', 'ab'],
 'num_col2': [1.1, 1.2, 1.3, 1.25, 1.7],
 'cat_col2': ['gh', 'gg', 'gg', 'gg', 'gg'],
 'cat_col3': ['mm', 'nn', 'aa', 'hh', 'jj']}
```

5️⃣`to_csv`

Purpose: Write the DataFrame to a CSV file.
Args:

- `path` (str, optional): Path to save the CSV. Default: `'data/out.csv'`.
  Returns:
- None. Writes the CSV file using `write_file` from file_handler module.