FREE EDITION · NOTES + 3 SAMPLE MCQS

CUET · COMPUTER SCIENCE · CLASS XII · CODE 308

Data Handling using Pandas - II

CUET unit: Data Handling using Pandas - II

By **UniDrill** · NCERT-grounded study material

WWW.UNIDRILL.IN

📌 Snapshot

- Pandas DataFrame operations extend to advanced statistical analysis, data aggregation, sorting, reshaping, and database connectivity — all high-frequency CUET topics.
- Descriptive statistics (max, min, mean, median, mode, variance, std, quartile) are tested both as direct recall ("which function returns the median?") and as code-output tracing.
- GROUP BY, pivot, and pivot_table are conceptually rich topics that NTA tests through scenario-based questions requiring students to predict output or identify the correct function.
- Handling missing values (NaN) — detecting with isnull(), dropping with dropna(), and filling with fillna() — is a dedicated exam-ready subtopic.
- The MySQL import/export functions (read_sql_query, read_sql_table, to_sql) test knowledge of the pymysql/sqlalchemy connection workflow.

📖 Detailed Notes

2.1 Core concepts

- **Introduction and Case Study:** Pandas is a Python library for data manipulation and analysis. A marks DataFrame (`marksUT`) of 4 students (Raman, Zuhaire, Ashravy, Mishti) across 3 unit tests and 5 subjects (Maths, Science, S.St, Hindi, Eng) is the running example throughout all sections. (NCERT §3.1, p. 63–64)
- **DataFrame.max():** Returns the maximum value of each column. By default includes all data types (Name column returns alphabetically maximum value). Pass `numeric_only=True` to restrict to numeric columns. Pass `axis=1` to get row-wise maximum. (NCERT §3.2.1, p. 65–66)
- **DataFrame.min():** Returns the minimum value of each column or row. Works identically to max() — supports `numeric_only=True` and `axis=1` parameters. (NCERT §3.2.2, p. 67)
- **DataFrame.sum():** Returns the sum of all values per column, regardless of datatype. For text columns it concatenates strings. To get the sum of a specific column, specify the column name: `df['Maths'].sum()` . For row-wise sum use `axis=1` . (NCERT §3.2.3, p. 68–69)

- **DataFrame.count():** Returns the total number of non-null values in each column (default axis=0). Use `axis=1` to count values per row. Unlike sum/mean, count works on all data types and does not skip NaN automatically — it counts only present values. (NCERT §3.2.4, p. 69–70)
- **DataFrame.mean():** Returns the mean (average) of numeric columns only. `df.mean()` gives column-wise averages; `axis=1` gives row-wise averages. (NCERT §3.2.5, p. 70–71)
- **DataFrame.median():** Returns the middle value of numeric data. When the count of values is even, the median is the average of the two middle values. Only applicable to numeric values. (NCERT §3.2.6, p. 71–72)
- **DataFrame.mode():** Returns the value that appears the most number of times in the data. The mode is defined as the most frequent value. Applicable per column. (NCERT §3.2.7, p. 72–73)
- **DataFrame.quantile():** Divides data into four parts. By default outputs the second quartile (median, q=0.5). Use `q=0.25` for Q1 (25th percentile) and `q=0.75` for Q3 (75th percentile). Can accept a list: `quantile([0.25, 0.75])`. (NCERT §3.2.8, p. 73–74)
- **DataFrame.var():** Returns the variance — the average of squared differences from the mean. Applicable to numeric columns only. (NCERT §3.2.9, p. 74–75)
- **DataFrame.std():** Returns standard deviation, calculated as the square root of variance. (NCERT §3.2.10, p. 75)
- **DataFrame.describe():** Displays all descriptive statistics (count, mean, std, min, 25%, 50%, 75%, max) in a single command for all numeric columns. (NCERT §3.2.10, p. 75)
- **Note on axis parameter:** In most Pandas statistical functions, `axis=0` (default) gives column-wise output and `axis=1` gives row-wise output. This is the reverse of most other Python operations where axis=0 is rows. (NCERT §3.2.1 Note, p. 67)
- **Data Aggregations (§3.3):** Aggregation transforms a dataset into a single numeric value. The `aggregate()` or `agg()` function applies one or more aggregate functions (max, min, sum, count, std, var) to columns. Multiple functions can be applied at once: `df.aggregate(['max','count'])`. The `axis` parameter works the same way. (NCERT §3.3, p. 75–76)
- **Sorting a DataFrame (§3.4):** `DataFrame.sort_values(by, axis=0, ascending=True)` arranges data in ascending or descending order. The `by` parameter specifies the column(s) to sort on. Sorting on multiple columns: if two rows have the same value for the first sort column, the second column is used as a tiebreaker. Default is ascending order. (NCERT §3.4, p. 77–78)

- **GROUP BY Functions (§3.5):** `DataFrame.GROUP BY(column)` splits data into groups based on a criterion. It follows a split-apply-combine strategy: (1) Split the DataFrame into groups, (2) Apply a function to each group, (3) Combine results into a new DataFrame. Key methods on a GroupBy object: `first()`, `size()`, `groups`, `get_group(name)`. Grouping by multiple columns is also possible. Aggregation functions can be applied on GroupBy objects using `agg()`. (NCERT §3.5, p. 79–82)
- **Altering the Index (§3.6):** By default, a numeric index starting from 0 is assigned. When a DataFrame is sliced, the index becomes non-continuous. `reset_index(inplace=True)` creates a new continuous index while preserving the original. The original index column can be dropped with `drop(columns=['index'], inplace=True)`. A column can be set as the index using `set_index('column_name', inplace=True)` and reverted using `reset_index('column_name', inplace=True)`. (NCERT §3.6, p. 82–84)
- **Reshaping Data — pivot() (§3.7.1A):** `df.pivot(index=, columns=, values=)` reshapes a DataFrame by using one column as the new row index, another as new column headers, and a third as cell values. Cells without matching entries are filled with NaN. Pivot fails with a ValueError if the index column has duplicate values. (NCERT §3.7.1A, p. 84–86)
- **Pivoting by Multiple Columns (§3.7.1B):** Multiple column names can be passed to the `values` parameter of `pivot()`. Omitting `values` will display all numeric columns in the pivoted form. (NCERT §3.7.1B, p. 86–87)
- **pivot_table() (§3.7.1C):** Works like `pivot()` but handles duplicate index entries by applying an aggregate function. Syntax: `pandas.pivot_table(data, values=None, index=None, columns=None, aggfunc='mean')`. The default aggregate function is mean. The `aggfunc` parameter can accept a list of functions or a dictionary mapping columns to different functions. (NCERT §3.7.1C, p. 88–89)
- **Handling Missing Values (§3.8):** A missing value is denoted by NaN. Two main strategies: (i) drop the row with missing value, (ii) fill or estimate the missing value. (NCERT §3.8, p. 89–90)
- **Checking Missing Values (§3.8.1):** `df.isnull()` returns a boolean DataFrame — True where value is missing. `df.isnull().any()` returns True for each column that has at least one missing value. `df.isnull().sum()` gives the count of NaN per column. `df.isnull().sum().sum()` gives total NaN count in the entire DataFrame. (NCERT §3.8.1, p. 91–93)
- **Dropping Missing Values (§3.8.2):** `df.dropna()` removes any row containing at least one NaN. Use `inplace=True` to modify the original DataFrame. `dropna(how='any')` is the default (drop row if any NaN present). Using dropna reduces dataset size, so it should be used sparingly. (NCERT §3.8.2, p. 94–96)
- **Estimating/Filling Missing Values (§3.8.3):** `df.fillna(num)` replaces all NaN with the specified value. `fillna(0)` replaces with 0, `fillna(1)` with 1.

`fillna(method='pad')` replaces a NaN with the value immediately before it (forward fill). `fillna(method='bfill')` replaces with the value immediately after it (backward fill). (NCERT §3.8.3, p. 96–97)

- **Import/Export between Pandas and MySQL (§3.9):** Requires two libraries: `pymysql` (database driver, install with `pip install pymysql`) and `sqlalchemy` (connection interface, install with `pip install sqlalchemy`). The `create_engine()` function from sqlalchemy establishes a connection using the connection string: `'mysql+pymysql://username:password@host:port/database_name'`. Default MySQL port is 3306. (NCERT §3.9, p. 98–99)
- **Importing from MySQL to Pandas (§3.9.1):** Three functions: (1) `pd.read_sql_query(query, sql_conn)` — reads SQL query result into DataFrame; (2) `pd.read_sql_table(table_name, sql_conn)` — reads an entire table into DataFrame; (3) `pd.read_sql(sql, sql_conn)` — reads either a query or a table name. (NCERT §3.9.1, p. 99–100)
- **Exporting from Pandas to MySQL (§3.9.2):** `DataFrame.to_sql(table, sql_conn, if_exists='fail', index=False/True)`. The `if_exists` parameter accepts: `'fail'` (default — raises ValueError if table exists), `'replace'` (overwrites existing table), `'append'` (adds rows to existing table). The `index` parameter controls whether the DataFrame index is written to MySQL. (NCERT §3.9.2, p. 100–101)

2.2 Definitions to memorise

Term	Definition	Page
Descriptive Statistics	Methods used to summarise data and get a basic idea about the dataset	65
max()	Returns the maximum value in each column (or row with axis=1) of a DataFrame	65
min()	Returns the minimum value in each column (or row with axis=1) of a DataFrame	67
sum()	Returns the sum of all values in each column of a DataFrame	68
count()	Returns the total number of non-null values in each column or row	69
mean()	Returns the arithmetic average of numeric values in each column	70
median()	Returns the middle value of sorted data; average of two middle values when count is even	71
mode()	Returns the value that appears the most number of times in the data	72
quantile()	Divides data into four parts; Q1=25%, Q2=50% (median), Q3=75%	73

Term	Definition	Page
variance (var())	Average of squared differences from the mean	74
Standard Deviation (std())	Square root of variance	75
describe()	Displays all descriptive statistics in a single command	75
Aggregation	Transforming a dataset to produce a single numeric value; functions: max, min, sum, count, std, var	75
sort_values()	Arranges DataFrame data in ascending or descending order by specified column(s)	77
GROUP BY	Splits data into groups based on criteria; follows split-apply-combine strategy	79
Altering the Index	Changing the row labels/indexes of a DataFrame using reset_index() or set_index()	82
Reshaping	Changing the shape (structure) of a DataFrame; done using pivot() or pivot_table()	84
pivot()	Reshapes DataFrame by using one column as index, another as column headers, and a third as values; fails with duplicate index entries	85
pivot_table()	Like pivot() but handles duplicate entries using an aggregate function (default: mean)	88
NaN	Not a Number — symbol used to denote a missing value in a DataFrame	89
isnull()	Returns a boolean DataFrame with True where values are missing (NaN)	91
dropna()	Removes entire rows containing any missing value from a DataFrame	94
fillna()	Replaces missing values with a specified number or using a method (pad/bfill)	96
pymysql	Python library (database driver) required to connect Python to MySQL	98
sqlalchemy	Python library that provides create_engine() for establishing MySQL connections	98
create_engine()	sqlalchemy function that establishes a connection to MySQL using a connection string	98
read_sql_query()	Reads result of an SQL query into a pandas DataFrame	99
read_sql_table()	Reads an entire MySQL table into a pandas DataFrame	99
to_sql()	Writes a pandas DataFrame to a MySQL table	100
Q1 (25th percentile)	First quartile — splits the lower 25% of data	73

Term	Definition	Page
Q3 (75th percentile)	Third quartile — splits the upper 25% of data	73
`numeric_only=True`	Argument restricting a statistical function to numeric columns	65
Forward fill (`pad`)	Fill NaN with the previous non-NaN value	96
Backward fill (`bfill`)	Fill NaN with the next non-NaN value	96
`set_index()`	DataFrame method that turns a column into the index	83
`reset_index()`	DataFrame method that creates a fresh integer index	82
split-apply-combine	The conceptual strategy behind GROUP BY	79
`agg()` / `aggregate()`	DataFrame method applying one or more aggregate functions	75-76
Default port (MySQL)	3306	98
Connection string	URL of form `mysql+pymysql://user:pwd@host:port/db`	98

2.3 Diagrams / processes to remember

- **Split-Apply-Combine diagram (Figure 3.1, p. 79):** Shows a two-column DataFrame (key, data) being split into groups A, B, C; sum applied to each group; results combined into a new DataFrame. This is the conceptual backbone of GROUP BY.
- **describe() output table (p. 75):** Shows all statistics (count, mean, std, min, 25%, 50%, 75%, max) for every numeric column side by side — the only function that gives a full statistical summary in one command.
- **pivot() vs pivot_table() difference (p. 85–88):** pivot() fails on duplicate index values (raises ValueError), while pivot_table() handles duplicates through aggregation (default: mean).
- **fillna() methods (p. 96–97):** `fillna(0)` replaces with zero; `fillna(method='pad')` replaces with the previous value; `fillna(method='bfill')` replaces with the next value — all three scenarios are tested in CUET.

2.4 Common confusions / NTA trap points

- **axis=0 vs axis=1 in statistical functions:** In Pandas statistical methods (max, min, mean, etc.), axis=0 (default) produces column-wise results and axis=1 produces row-wise results. NTA distractors often swap these — "axis=1 gives column-wise output" is a classic wrong option.
- **pivot() vs pivot_table():** Students confuse when to use each. Key rule: if index column has duplicate values, pivot() raises ValueError and pivot_table() must be used. The default aggfunc in pivot_table() is mean, not sum.

- **dropna() vs fillna():** dropna() removes the entire row (reduces dataset size), while fillna() preserves the row and substitutes a value. NTA sometimes presents scenarios where the wrong strategy is applied.
- **isnull() vs any():** `df.isnull()` returns a full boolean DataFrame; `df.isnull().any()` returns one True/False per column; `df.isnull().sum()` returns count of NaN per column. Confusing these in code-output questions is a frequent error.
- **to_sql() if_exists parameter (NCERT §3.9.2, p. 100-101).** 'fail' (default), 'replace', 'append'. 'overwrite' does NOT exist.
- `pivot_table` **default aggfunc is** `mean` **(NCERT §3.7.1C, p. 88).** Not sum. NTA distractor: claims sum is default.
- `mode()` **returns a DataFrame, not a single value (NCERT §3.2.7, p. 72-73).** Because data may be multimodal.
- `std()` **≠** `var()` **(NCERT §3.2.9-10, p. 74-75).** std is square root of variance.
- `quantile()` **defaults to 0.5 (median) (NCERT §3.2.8, p. 73-74).**
- `describe()` **only summarises numeric columns (NCERT §3.2.10, p. 75).**
- `read_sql_query` **vs** `read_sql_table` **(NCERT §3.9.1, p. 99).** Former takes a query string; latter takes a table name.

🎯 Practice MCQs

Q1. Which of the following Pandas functions displays all descriptive statistical values (count, mean, std, min, quartiles, max) for all numeric columns in a single command?

A. df.aggregate()

B. df.summary()

C. df.describe()

D. df.stat()

Q2. Consider a DataFrame `df`. What will `df.max(axis=1)` return?

A. The maximum value of each column across all rows

B. The maximum value of each row across all columns

C. The maximum value of the entire DataFrame as a single scalar

D. An error, because axis=1 is not valid for max()

Q3. Which of the following statements about `DataFrame.mode()` is correct?

A. It returns the arithmetic average of values in each column.

B. It returns the middle value when data is sorted in ascending order.

C. It returns the value that appears the most number of times in the data.

D. It is applicable only to numeric columns and ignores text values.

📊 PYQ Alignment

DataFrame statistics and aggregation are heavily tested in CUET (UG) Computer Science/ Informatics Practices papers, with questions almost every year on descriptive statistics functions (especially mean, median, mode, and the axis parameter), GROUP BY operations, and missing value handling. Case/code-based questions tracing the output of sort_values(), pivot_table(), or fillna() with method='pad'/'bfill' are particularly common, reflecting the emphasis on practical DataFrame manipulation. See PYQ archive for Computer Science.