Pandas -Create

Series	`s = pd.Series([1, 2, 3], index=['a', 'b', 'c'])`	One-dimensional labeled array (like a column).
DataFrame	`df = pd.DataFrame({'A':[1,2], 'B':[3,4]})`	Two-dimensional table (rows × columns) with labels.
Panel *(deprecated)*	–	3-D data (replaced by `xarray` or MultiIndex).

Input

Command	Example	Purpose
`pd.DataFrame()`	`pd.DataFrame(data, index=None, columns=None)`	Create a DataFrame from dicts, arrays, or another DataFrame.
`pd.Series()`	`pd.Series(data, index=None)`	Create a Series.
`pd.read_csv()`	`pd.read_csv(filepath_or_buffer, sep=',', ...)`	Read a CSV file into a DataFrame.
`pd.read_excel()`	`pd.read_excel(io, sheet_name=0, ...)`	Read an Excel file.
`pd.read_sql()`	`pd.read_sql(sql, con, ...)`	Read SQL query into a DataFrame.
`pd.read_json()`	`pd.read_json(path_or_buf, ...)`	Read JSON.
`pd.read_html()`	`pd.read_html(io, match='table', ...)`	Scrape tables from HTML.
`pd.read_clipboard()`	`pd.read_clipboard(sep='\t', ...)`	Read data from the clipboard.
`pd.read_parquet()`	`pd.read_parquet(path, ...)`	Read Parquet file (columnar).
`pd.DataFrame.from_records()`	`DataFrame.from_records(array, index=None)`	Build from list of tuples/dicts.
`pd.DataFrame.from_dict()`	`DataFrame.from_dict(data, orient='columns')`	Build from dict of lists/Series.

index-slice-filter

Command	Example	Purpose
Single column	`df['col']` or `df.col`	Returns a Series.
Multiple columns	`df[['a', 'b']]`	Returns a new DataFrame.
Row by label	`df.loc['row_label']`	Access rows by index label.
Row by position	`df.iloc[3]`	Access the 4th row (integer index).
Conditional filtering	`df[df['A'] > 10]`	Return rows where condition is True.
Boolean indexing	`mask = df['A'] > 10; df[mask]`	Same as above, but mask reusable.
`.at[]` / `.iat[]`	`df.at['row', 'col']` or `df.iat[3, 1]`	Fast scalar access.
`.loc[]` with slices	`df.loc['row1':'row5', 'A':'C']`	Label-based slice.
`.iloc[]` with slices	`df.iloc[0:5, 1:3]`	Integer slice.
`.xs()`	`df.xs('label', axis=0, level='level_name')`	Cross-section for MultiIndex.
`.head(n)` / `.tail(n)`	`df.head(10)`	First/last n rows.
`.sample(n)`	`df.sample(5, random_state=0)`	Random rows.
`.set_index()`	`df.set_index('col')`	Make a column the index.
`.reset_index()`	`df.reset_index(drop=True)`	Drop the index back to default.

Modification-assignment

Operation	Syntax	What it does
Assign new column	`df['new'] = df['A'] + df['B']`	Add or overwrite column.
`.assign()`	`df.assign(new=lambda x: x.A + x.B)`	Chainable column creation.
`.loc[]` assignment	`df.loc[df['A'] > 10, 'B'] = 0`	Conditional update.
`.replace()`	`df.replace(to_replace=5, value=0)`	Replace values.
`.fillna()`	`df.fillna(method='ffill')`	Forward-fill NaNs.
`.drop()`	`df.drop(columns=['C'], axis=1)`	Remove rows/columns.
`.dropna()`	`df.dropna(subset=['A', 'B'])`	Drop rows with NaNs in specified columns.
`.rename()`	`df.rename(columns={'A':'alpha'}, inplace=True)`	Rename columns/indices.
`.insert()`	`df.insert(0, 'first', df['B']*2)`	Insert column at position.
`.pop()`	`col = df.pop('B')`	Remove and return column.
`.astype()`	`df['A'] = df['A'].astype('float64')`	Convert dtype.
`.copy()`	`df2 = df.copy(deep=True)`	Deep copy of DataFrame.

Aggregation - Grouping

Function	Syntax	What it does
`.sum()`	`df['A'].sum()`	Sum of a column/axis.
`.mean()`	`df.mean()`	Mean value.
`.median()`	`df.median()`	Median.
`.min()`, `.max()`	`df.min()` / `df.max()`	Min/Max.
`.std()`, `.var()`	`df.std()` / `df.var()`	Standard deviation / variance.
`.describe()`	`df.describe()`	Summary statistics (count, mean, std, min, 25%, 50%, 75%, max).
`.groupby()`	`df.groupby('col').agg({'A':'sum', 'B':'mean'})`	Group by one or more keys and aggregate.
`.pivot_table()`	`pd.pivot_table(df, values='val', index='row', col`	Create pivot table.
`.crosstab()`	`pd.crosstab(df['A'], df['B'])`	Cross-tabulation of two factors.
`.value_counts()`	`df['A'].value_counts()`	Frequency of each value.
`.corr()`, `.cov()`	`df.corr(method='pearson')` / `df.cov()`	Correlation / covariance matrix.
`.quantile(q)`	`df.quantile(0.25)`	25th percentile, etc.
`.apply()`	`df['A'].apply(np.sqrt)`	Apply a function element-wise.
`.transform()`	`df.groupby('grp')['A'].transform('mean')`	Transform within groups.
`.applymap()`	`df.applymap(lambda x: x*2)`	Element-wise on DataFrame.

Merge-Concat

Function	Syntax	What it does
`.merge()`	`pd.merge(df1, df2, on='key', how='inner')`	SQL-style join.
`.join()`	`df1.join(df2, on='key', how='left')`	Join on index or column.
`.concat()`	`pd.concat([df1, df2], axis=0)`	Concatenate vertically or horizontally.
`.append()`	`df1.append(df2, ignore_index=True)`	Append rows (deprecated → use concat).
`.cross_join()` (new in 1.2)	`df1.merge(df2, how='cross')`	Cartesian product.
`.merge_asof()`	`pd.merge_asof(df1, df2, on='time', direction='nea`	Merge nearest key (useful for time series).
`.merge_ordered()`	`pd.merge_ordered(df1, df2, on='date')`	Merge with ordered keys (keeps missing values).

Time-Date

Function	Syntax	What it does
`.to_datetime()`	`pd.to_datetime(df['date_str'])`	Parse strings to datetime.
`.to_timedelta()`	`pd.to_timedelta(df['duration_str'])`	Parse time deltas.
`.date_range()`	`pd.date_range(start='2024-01-01', periods=10, fre`	Generate a sequence of dates.
`.resample()`	`df.resample('M').mean()`	Resample time series to a new frequency.
`.asfreq()`	`df.asfreq('D')`	Change frequency without aggregation.
`.shift()`	`df['lag1'] = df['value'].shift(1)`	Lag or lead columns.
`.rolling()`	`df['value'].rolling(window=3).mean()`	Moving window calculations.
`.expanding()`	`df['value'].expanding().sum()`	Expanding window (cumulative).
`.ewm()`	`df['value'].ewm(span=10, adjust=False).mean()`	Exponentially weighted functions.
`.dt` accessor	`df['date'].dt.month`	Extract components (year, month, day, etc.).
`.tz_localize()` / `.tz_convert()`	`df['date'].dt.tz_localize('UTC')` / `.tz_convert('US`	Time-zone handling.

Missing-Data

Function	Syntax	What it does
`.isna()` / `.notna()`	`df.isna()`	Boolean mask of missing values.
`.dropna()`	`df.dropna(axis=0, how='any', subset=['A'])`	Remove rows/columns with NaNs.
`.fillna()`	`df.fillna(method='ffill')`	Forward/backward fill or specific value.
`.interpolate()`	`df.interpolate(method='linear')`	Fill NaNs by interpolation.
`.replace()`	`df.replace(to_replace='?', value=np.nan)`	Convert sentinel values to NaN.

Save-File

Function	Syntax	What it does
`.to_csv()`	`df.to_csv('out.csv', index=False)`	Write to CSV.
`.to_excel()`	`df.to_excel('out.xlsx', sheet_name='Sheet1')`	Write to Excel.
`.to_sql()`	`df.to_sql(name='table', con=engine, if_exists='re`	Write to SQL database.
`.to_json()`	`df.to_json('out.json', orient='records')`	Write JSON.
`.to_parquet()`	`df.to_parquet('out.parquet')`	Write Parquet.
`.to_html()`	`df.to_html('out.html')`	Write as an HTML table.
`.to_clipboard()`	`df.to_clipboard(index=False)`	Copy to clipboard.

Other-Commands

Function	Syntax	What it does
`.info()`	`df.info(verbose=True)`	Compact summary (dtype, non-null count).
`.head()` / `.tail()`	`df.head(5)`	Preview rows.
`.sample()`	`df.sample(frac=0.1)`	Random sample of rows/columns.
`.query()`	`df.query('A > 10 and B < 5')`	SQL-like query on DataFrame.
`.eval()`	`df.eval('C = A + B')`	Evaluate string expressions.
`.describe()`	`df.describe(include='all')`	Summary statistics.
`.memory_usage()`	`df.memory_usage(deep=True)`	Memory footprint of each column.
`.copy()`	`df2 = df.copy(deep=True)`	Make a copy.
`.to_dict()`	`df.to_dict(orient='records')`	Convert to dict/list.
`.stack()` / `.unstack()`	`df.stack()`	Convert columns to rows (MultiIndex).
`.melt()`	`pd.melt(df, id_vars=['id'], var_name='variable',`	Unpivot DataFrame.
`.wide_to_long()`	`pd.wide_to_long(df, stubnames=['A', 'B'], i='id',`	Convert wide to long format.
`.get_dummies()`	`pd.get_dummies(df['category'])`	One-hot encode.
`.cut()` / `.qcut()`	`pd.cut(df['A'], bins=5)`	Bin continuous values.
`.rank()`	`df['A'].rank(method='average')`	Rank values.
`.corrwith()`	`df.corrwith(df2)`	Correlation with another Series/DataFrame.