Pandas Cheat Sheet: Beginner to Advanced

1. Import and Basic Structures

```
import pandas as pd
```

Task	Example	Explanation
Create Series	`s = pd.Series()`	1D labeled array
Create DataFrame	`df = pd.DataFrame({'A':,'B':})`	2D labeled table, cols as dict of lists
Read CSV/Excel	`pd.read_csv('file.csv')` `pd.read_excel('file.xlsx')`	Load data from file
Write CSV/Excel	`df.to_csv('out.csv')` `df.to_excel('out.xlsx')`	Save dataframe to file
Quick view	`df.head()`, `df.tail()`, `df.info()`, `df.describe()`	See data summary/statistics

2. Selection, Indexing & Slicing

Task	Example	Explanation
Select column	`df['A']`	Get Series for column 'A'
Select multiple cols	`df[['A','B']]`	List of columns
Row by position/label	`df.iloc`, `df.loc`	Integer/label selection
Slice rows	`df[1:4]`	Like Python list slicing
Boolean mask filtering	`df[df['A'] > 2]`	Filter rows with condition
Query string	`df.query('A > 2 & B < 5')`	SQL-like querying
Set index / Reset index	`df.set_index('A')`, `df.reset_index()`	Change row labels

3. Data Cleaning & Transformation

Task	Example	Explanation
Rename columns	`df.rename(columns={'A':'alpha'})`	Change column names

Drop rows/cols	`df.drop('A', axis=1)`, `df.drop(0, axis=0)`	Remove column/row
Fill missing	`df.fillna(0)`	Replace NaN with value
Drop missing	`df.dropna()`	Remove rows with missing
Replace values	`df.replace({1: 10})`	Substitute values
Type conversion	`df['B'] = df['B'].astype(float)`	Change dtype
String ops	`df['col'].str.lower()`, `str.contains('x')`	Vectorized string functions
Apply function	`df['A'].apply(np.log)`, `df.apply(func, axis=1)`	Per-row or per-col computation
Lambda	`df['A'].apply(lambda x: x+1)`	Inline anonymous function
Map (Series)	`df['A'].map({1:2, 2:3})`	Substitute using dict/function

4. Grouping, Aggregation & Summarization

Task	Example	Explanation
Group by & aggregate	`df.groupby('A').sum()`	Sum for each unique 'A' value
Multiple aggregations	`df.groupby('A').agg({'B':['mean','max']})`	Several stats for group
Pivot table	`df.pivot_table(index='A', columns='B', values='C', aggfunc='mean')`	Summarize data
Count values	`df['A'].value_counts()`	Frequency of each unique value
Crosstab (frequency table)	`pd.crosstab(df['A'], df['B'])`	Count matrix by two categories

5. Merging, Joining & Concatenation

Task	Example	Explanation

Concatenate (vertically/horiz.)	`pd.concat([df1, df2], axis=0)`
	`pd.concat([df1, df2], axis=1)`
		Stack rows or columns

Merge (SQL join)	`pd.merge(df1, df2, on='A', how='left')`	Join on key columns, left/right/inner/outer
Join by index	`df1.join(df2, how='inner')`	Use row labels for join

6. Dates, Times, Categoricals

Task	Example	Explanation
Convert to datetime	`pd.to_datetime(df['date_col'])`	Parse strings as dates
Datetime accessor	`df['date'].dt.year`, `.dt.month`	Extract year, month, weekday, etc.
Resample by time interval	`df.resample('M').mean()`	Downsample by month, compute mean
Categoricals (memory, speed)	`df['A'] = df['A'].astype('category')`	Categorical/ordinal types

7. Advanced Indexing & MultiIndex

Task	Example	Explanation
Multi-level index	`df.set_index(['A','B'])`	Index by more than one column
Access multi-index	`df.loc[('foo',1), :]`	Select rows with multiple index keys
Stack/Unstack	`df.stack()`, `df.unstack()`	Pivot cols to rows or vice versa
Swap index levels	`df.swaplevel()`	Change order of multi-indices

8. Windowing, Rolling & Expanding

Task	Example	Explanation
Rolling window	`df['A'].rolling(3).mean()`	Moving average (window=3)
Expanding window	`df['A'].expanding().sum()`	Cumulative sum over all previous rows
Exponential Weighted	`df['A'].ewm(span=3).mean()`	EWMA—for smoothing/noisy time series

9. Visualization

Simple, built-in plotting using matplotlib as backend:

```
df.plot(kind='line')          # Line plot (default)
df.plot(kind='bar')           # Bar plot
df['col'].hist()              # Histogram
df.plot.scatter(x='A', y='B')  # Scatter plot
```

10. Performance, Memory, Efficiency

Task	Example	Explanation
Categorical dtype	`df['col'] = df['col'].astype('category')`	Memory savings
Chunked processing	`pd.read_csv('file.csv', chunksize=10000)`	Process large files by chunk
Vectorization	`df['A'] + df['B']`	Use vector ops, avoid Python loops
Profiling	`df.memory_usage(deep=True)`	See memory usage

11. Integration

Task	Example	Explanation
NumPy interoperability	`df.values`, `df.to_numpy()`	Convert DataFrame to array
sklearn	`from sklearn.preprocessing import StandardScaler`	pandas DataFrames work as input to sklearn
Export to Excel/CSV	`df.to_csv(...)`, `df.to_excel(...)`	Easy saving/sharing

12. Pandas Project Workflow Example

```
import pandas as pd

# Load data
df = pd.read_csv('data.csv')

# Clean & explore
df = df.dropna()
df['target'] = df['target'].astype('category')
df.describe(), df.info()
```

```
# Feature engineering
df['age_group'] = pd.cut(df['age'], bins=[0,18,65,120],
labels=['child','adult','senior'])

# Group & summarize
grouped = df.groupby('age_group')['salary'].mean()

# Merge more data
df_extra = pd.read_csv('extra.csv')
merged = pd.merge(df, df_extra, on='id', how='left')

# Save results
merged.to_csv('processed.csv', index=False)
```

13. Advanced: Method Chaining (Pipelines)

Elegant "pipe" syntax for complex processing:

```
result = (
    df
    .dropna(subset=['income'])
    .assign(income_log = lambda x: np.log1p(x['income']))
    .groupby('group')['income_log']
    .mean()
    .reset_index()
)
```

14. Debugging, Learning, and Documentation

- Get help/doc: `pd.Series?`, `df.method?`, `help(pd.concat)`
- Run `df.sample(5)` to peek at random rows
- For very large DataFrames: use `.info(memory_usage='deep')`, and try `dask` for out-of-core processing