📝 Master Data Science & Pandas EDA Cheatsheet

---

1. Data Loading

Task	Function / Notes
Read CSV	`df = pd.read_csv('filename.csv')`
Read Excel	`df = pd.read_excel('filename.xlsx')`
Read SQL	`df = pd.read_sql(query, connection)`
Read JSON	`df = pd.read_json('filename.json')`
Read XML	`df = pd.read_xml('filename.xml')`
Read HTML Table	`dfs = pd.read_html('url')`
Chunking Large Files	`pd.read_csv('file.csv', chunksize=1000)`

---

2. Basic Data Inspection

Task	Function / Notes
Top / Bottom rows	`df.head()` `df.tail()`
Data Types	`df.dtypes`
Summary Statistics	`df.describe()`

Info / Columns / Index	`df.info()`
	`df.columns`
	`df.index`

Check Missing	`df.isna().sum()` `df.isnull().sum()`

Unique / Counts	`df['col'].unique()`
	`df['col'].nunique()`
	`df['col'].value_counts()`

Memory Usage	`df.memory_usage(deep=True)`

---

3. Data Cleaning & Missing Data

Task	Function / Notes
Drop Rows/Columns	`df.drop(columns=['col'])` `df.dropna()`

Fill Missing	`df.fillna(value)`
	`df['col'].ffill()`
	`df['col'].bfill()`
	`df['col'].interpolate()`

Fill by Group	`df['col'] = df.groupby('group')['col'].transform(lambda x: x.fillna(x.median()))`
Conditional Fill	`df['col'] = df['col'].apply(lambda x: 0 if pd.isna(x) else x)`
Rename Columns	`df.rename(columns={'old':'new'})`
Remove Duplicates	`df.drop_duplicates(subset=['col1','col2'], keep='first')`
Assert / Validation	`assert df.notnull().all().all(), "Missing values!"`

---

4. Data Transformation

Task	Function / Notes
Apply Function / Custom	`df['col'].apply(lambda x: func(x))`
Apply Across Rows	`df.apply(lambda row: func(row['col1'], row['col2']), axis=1)`
GroupBy & Aggregate	`df.groupby('col').agg({'col2':'sum'})`
Multiple Aggregate	`df.groupby('col').agg(['mean','sum'])`

Transform Function	`df.groupby('col').transform(lambda x: x-x.mean())`
Pivot Table	`df.pivot_table(index='col1', columns='col2', values='col3', aggfunc='sum', fill_value=0)`

Melt / Stack / Unstack	`df.melt(id_vars=['col1'])`
	`df.stack()`
	`df.unstack()`

Cross-tab	`pd.crosstab(df['col1'], df['col2'])`
Merge / Join	`pd.merge(df1, df2, on='col', how='left')`
Concatenate	`pd.concat([df1, df2], axis=0)`

---

5. Indexing & Selection

Task	**Function / Notes**
Select Columns	`df['col']` `df[['col1','col2']]`
Select Rows by Position	`df.iloc[0:5]`
Select Rows by Label	`df.loc[0:5]`
Conditional Selection	`df[df['col']>value]`
Query Function	`df.query('col>value')`
Filtering with isin	`df[df['col'].isin([v1,v2])]`
Reset / Set Index	`df.reset_index(drop=True)` `df.set_index('col')`

MultiIndex	`df.set_index(['col1','col2'])`
	`df.xs(key, level='level_name')`
	`df.loc[(slice(start1, end1), slice(start2, end2)), :]`

---

6. Data Formatting & Conversion

Task	Function / Notes
Convert Type	`df['col'].astype('type')`
Datetime Conversion	`pd.to_datetime(df['col'], format='%Y-%m-%d', errors='coerce')`

String Operations	`df['col'].str.lower()`
	`df['col'].str.upper()`
	`df['col'].str.split()`
	`df['col'].str.contains()`
	`df['col'].str.extract(r'(regex)')`

Categorical	`df['col'].astype('category')` `df['col'].cat.set_categories(['cat1','cat2'], ordered=True)`
Numeric Conversion	`df['col'] = pd.to_numeric(df['col'], errors='coerce')`
Mixed-Type Cleanup	`df['col'] = df['col'].astype(str).str.replace('$','').astype(float)`

---

7. Handling Time Series

Task	Function / Notes
Set Datetime Index	`df.set_index(pd.to_datetime(df['date_col']))`

Resample	`df.resample('frequency').agg_func()`
	`'D' → Daily`
	`'W' → Weekly`
	`'M' → Monthly`
	`'Q' → Quarterly`
	`'Y' → Yearly`
	`agg_func() => sum(), mean(), max()`

Rolling Operations	`df.rolling(window=5).mean()`
Lag / Shift	`df['col'].shift(1)`
Time-Based Groupby	`df.groupby(pd.Grouper(key='date_col', freq='M')).sum()`

Missing Timestamps	`df = df.asfreq('D')`

---

8. Numeric Transformations / Normalization

Task	Function / Notes
Min-Max Normalization	`(df['col']-df['col'].min())/(df['col'].max()-df['col'].min())`
Z-score Standardization	`(df['col']-df['col'].mean())/df['col'].std()`
Round Values	`df['col'].round(2)`

Aggregate Functions	`df.sum()`
	`df.mean()`
	`df.median()`
	`df.std()`

---

9. Visualization

Task	Function / Notes
Histogram	`df['col'].hist(bins=20)` `sns.histplot(df['col'])`
Boxplot	`df.boxplot(column=['col1','col2'])` `sns.boxplot(x='cat', y='num', data=df)`
Scatter	`df.plot.scatter(x='col1', y='col2')` `sns.scatterplot(x='col1', y='col2')`
Line	`df.plot.line()` `sns.lineplot(x='x', y='y')`
Bar	`df['col'].value_counts().plot.bar()` `sns.barplot(data=df, x='col1', y='col2', hue='col3')`
Pairplot / Correlation	`sns.pairplot(df)` `sns.heatmap(df.corr(), annot=True)`

Matplotlib Customization	`plt.style.use('ggplot')` `plt.show()`

---

10. File Export

Task	Function / Notes
CSV	`df.to_csv('filename.csv', index=False)`
Excel	`df.to_excel('filename.xlsx', index=False)`
JSON	`df.to_json('filename.json')`
SQL	`df.to_sql('table_name', connection)`

---

11. Performance & Large Data Handling

Task	Function / Notes
Reduce Memory	`df['col'] = pd.to_numeric(df['col'], downcast='float')`
Categorical Optimization	`df['cat_col'] = df['cat_col'].astype('category')`
Swifter	`df['col'].swifter.apply(lambda x: func(x))`
Dask for Parallel	`df = dd.from_pandas(df, npartitions=10)`
Sampling	`df.sample(n=1000)`

---

12. Statistical Analysis & SciPy / StatsModels

Task	Function / Notes
Correlation / Covariance	`df.corr()` `df.cov()`
Skewness / Kurtosis	`from scipy.stats import skew, kurtosis;` `skew(df['col']), kurtosis(df['col'])`
Linear Regression	`import statsmodels.api as sm;`

	`sm.OLS(y,X).fit()`

---

13. Text / String Data

Task	Function / Notes
String contains	`df[df['col'].str.contains('text')]`
String split	`df['col'].str.split(' ', expand=True)`
Regex extract	`df['col'].str.extract(r'(regex)')`

---

14. Outlier Detection & Handling

Task	Function / Notes
IQR Method	`Q1, Q3 = df['col'].quantile([0.25,0.75]); IQR=Q3-Q1; df[(df['col']>=Q1-1.5*IQR) & (df['col']<=Q3+1.5*IQR)]`
Z-score	`from scipy.stats import zscore; df[(np.abs(zscore(df['col']))<3)]`