Pandas Cheat Sheet

1. Setup & Structures

```
import pandas as pd
import seaborn as sns # For sample datasets
```

Core Objects

- **Series:** 1D array-like object (a single column).
- **DataFrame:** 2D matrix with labeled rows and columns.

2. Creating & Loading Data

Manual Creation

```
data = {
    "name": ["rahul", "vidu"],
    "marks": [100, 98]
}

# Default Index (0, 1, 2...)
df = pd.DataFrame(data)

# Custom Index
df = pd.DataFrame(data, index=[90, 91])
```

Loading Files

```
# CSV (with specific column as index)
df = pd.read_csv('data.csv', index_col="Index")

# JSON
df = pd.read_json('data.json')

# Sample Datasets (Seaborn)
df = sns.load_dataset('iris')    # 'titanic', 'tips'
```

Web Scraping (`read_html`) *Technique: Use `storage_options` to mimic a real browser if you get 403 errors.*

```
url = 'https://en.wikipedia.org/wiki/...'
tables = pd.read_html(
    url,
    storage_options={'User-Agent': 'Mozilla/5.0'}
)
df = tables[0] # Usually the first table
```

3. Inspection & Metadata

Method	Description
`.shape`	Dimensions (rows, cols)
`.columns`	List of column names
`.index`	List/Range of row labels
`.values`	Underlying NumPy array
`.info()`	Data types & non-null counts (Crucial check)
`.describe()`	Summary stats (mean, std, min, max)
`.value_counts()`	Frequency of unique values in a column

```
df.head(5)    # First n rows
df.tail(5)    # Last n rows
df.sample(5)  # Random n rows
```

4. Indexing & Selection

Column Selection

```
df['City']              # Returns Series
df[['City', 'Name']]    # Returns DataFrame
```

The `.loc` vs `.iloc` Matrix

Feature	`.loc[row_label, col_label]`	`.iloc[row_pos, col_pos]`
Logic	**Label** based (Names)	**Integer** based (Position)
Slicing	**Inclusive** (`0:2` includes 2)	**Exclusive** (`0:2` excludes 2)
Examples	`df.loc[90, 'name']`	`df.iloc[0, 0]`

Feature	`.loc[row_label, col_label]`	`.iloc[row_pos, col_pos]`
	`df.loc[0:2, 'name':'age']`	`df.iloc[:, 0:2]`

5. Filtering & Querying

Boolean Masking

```
# Step 1: Create Mask (True/False)
mask = df['Population'] < 10_000

# Step 2: Apply Mask
filtered = df[mask]

# Combined Conditions (& for AND, | for OR)
mask2 = (df['Population'] < 10k) & (df['Region'] == 'Americas')
df[mask2]
```

Query Syntax *SQL-like string syntax for cleaner code.*

```
df.query('Country == "Solomon Islands" and City == "Frankchester"')
```

6. Data Cleaning (Handling Nulls)

Detection

```
df.isnull().sum()   # Count missing values per column
```

Removal

```
df.dropna()         # Drop rows with ANY nulls
# Note: Use inplace=True to modify original df
```

Imputation (Filling)

```
df.fillna(0)                     # Fill all nulls with 0
df['name'].fillna('Unknown')     # Fill specific column

# Advanced: Dictionary filling (Different value per col)
```

```
df.fillna({
    "age": 20,
    "height": 150
})

# Advanced: Fill with Mean
df.fillna({
    "age": df['age'].mean(),
    "height": df['height'].mean()
})
```

7. Transformation & Data Types

Feature Engineering

```
df['power'] = 10                 # Assign scalar (broadcasts to all rows)
df['bmi'] = df['wt'] / df['ht']  # Vectorized math between columns
```

Data Types

```
df.dtypes                        # View types
df.select_dtypes(include=['number'])  # Get only numeric cols
df.select_dtypes(include=['object'])  # Get only text/categorical cols
```

8. Aggregation (GroupBy)

Split-Apply-Combine strategy.

```
grouped = df.groupby('place')

# Inspection
grouped.groups.keys()            # View group labels
grouped.get_group('bhadohi')     # Extract specific group DataFrame

# Aggregation
grouped['salary'].mean()         # Mean salary per place

# Iteration
for name, group_df in grouped:
    print(name)
    print(group_df)
```