Pandas Cheat Sheet (Tiered Approach)

Tier 1 Core Pandas (80% of Projects)

1. Data Loading/Saving

```
import pandas as pd
df = pd.read_csv('data.csv')          # Read CSV
df = pd.read_excel('data.xlsx')       # Read Excel
df.to_csv('output.csv', index=False)  # Save to CSV
```

2. DataFrame Basics

```
df.head(3)       # First 3 rows
df.info()        # Summary (dtypes, non-null counts)
df.describe()    # Stats for numeric columns
df.shape         # (rows, columns)
df.columns       # List all columns
```

3. Filtering & Selection

```
df['column']                  # Single column
df[['col1', 'col2']]          # Multiple columns
df.loc[0:5, 'col1':'col3']    # Label-based selection
df.iloc[0:5, 1:3]             # Position-based selection
df[df['age'] > 30]            # Boolean filtering
```

4. Sorting

```
df.sort_values('column', ascending=False)  # Sort by column
df.sort_index()                            # Sort by index
```

5. Missing Data

```
df.isnull().sum()          # Count missing values
df.dropna()                # Drop rows with NaN
df.fillna(0)               # Fill NaN with 0
```

6. Aggregation & Grouping

```
df.groupby('category')['value'].mean()  # Group and average
df.agg({'col1': 'sum', 'col2': 'mean'}) # Multiple aggregations
```

7. Column Operations

```
df['new_col'] = df['old_col'] * 2  # Create new column
df.drop('column', axis=1)          # Drop column
df.rename(columns={'old':'new'})   # Rename column
df['col'].astype('int')            # Convert dtype
```

Tier 2 Intermediate (Analytics & ML)

1. Merging & Joining

```
pd.merge(df1, df2, on='key')     # SQL-like join
pd.concat([df1, df2], axis=0)    # Stack vertically
df1.join(df2, how='left')        # Join on index
```

2. Datetime Handling

```
df['date'] = pd.to_datetime(df['date'])  # Convert to datetime
df['year'] = df['date'].dt.year          # Extract year
df.resample('M').sum()                   # Monthly resampling
```

3. Text Operations

```
df['text'].str.contains('abc')   # Filter text
df['text'].str.replace('old','new')  # Replace substring
df['text'].str.split(' ')        # Split into list
```

4. Advanced Grouping

```
df.groupby('group')['value'].transform('mean')  # Group-transform
df.pivot_table(index='a', columns='b', values='c', aggfunc='mean')
```

Tier 3 Advanced (Optimization & Scaling)

1. Performance Tricks

```
df.memory_usage(deep=True)  # Check memory usage
df['col'] = df['col'].astype('category')  # Reduce memory
```

2. Window Functions

```
df['rolling_avg'] = df['value'].rolling(3).mean()  # Rolling average
df['expanding_sum'] = df['value'].expanding().sum()
```

3. MultiIndex & Reshaping

```
df.set_index(['col1', 'col2'])  # Hierarchical index
pd.melt(df, id_vars=['id'])     # Wide to long format
```

Tier 1 Core Enhancements

Value Counts & Unique

```
df['col'].value_counts()    # Frequency of values
df['col'].unique()          # Unique values
```

Conditional Assignment

```
df['flag'] = df['col'] > 100
df.loc[df['col'] > 100, 'new_col'] = 'High'
```

Tier 2 Useful Additions

Apply & Lambda Functions

```
df['col2'] = df['col'].apply(lambda x: x * 2)
```

Duplicated Handling

```
df.duplicated()             # Check for duplicates
df.drop_duplicates()        # Remove duplicates
```

Tier 3 Useful Advanced Ops

Categorical Handling

```
df['col'] = pd.Categorical(df['col'], categories=['low', 'medium', 'high'],
ordered=True)
```

Query API

```
df.query('age > 30 and salary < 70000')
```

Chaining with Pipe

```
def clean_data(df): return df.dropna().reset_index(drop=True)
df = df.pipe(clean_data)
```