Pandas Data Analysis and Manipulation Cheat Sheet

|  |  |  |
| --- | --- | --- |
| Category | Operation | Code Example |
| Category | Operation | Code Example |
| Importing Data | Read CSV | df = pd.read\_csv('file.csv') |
|  | Read Excel | df = pd.read\_excel('file.xlsx') |
|  | Read from SQL Database | df = pd.read\_sql('SQL\_QUERY', connection) |
| Basic DataFrame Operations | View first 5 rows | df.head() |
|  | View last 5 rows | df.tail() |
|  | Get DataFrame shape (rows, columns) | df.shape |
|  | Summary statistics | df.describe() |
| Accessing Data | Access single column | df['Column'] |
|  | Access multiple columns | df[['Col1', 'Col2']] |
|  | Access row by index (position) | df.iloc[2] |
|  | Access row by label (index label) | df.loc[1] |
|  | Access specific cell (row, column) | df.loc[1, 'Column'] |
| Filtering Data | Filter by condition (e.g., Age > 30) | df[df['Age'] > 30] |
|  | Filter with multiple conditions | df[(df['Age'] > 30) & (df['City'] == 'New York')] |
| Sorting Data | Sort by column (ascending) | df.sort\_values('Age') |
|  | Sort by column (descending) | df.sort\_values('Age', ascending=False) |
| Handling Missing Data | Fill missing values with a constant | df.fillna(0, inplace=True) |
|  | Drop rows with missing values | df.dropna(inplace=True) |
| Grouping Data | Group by column and aggregate | df.groupby('City').mean() |
|  | Group by multiple columns | df.groupby(['City', 'Age']).sum() |
| Adding/Modifying Columns | Add a new column | df['Salary'] = [50000, 60000] |
|  | Modify an existing column | df['Age'] = df['Age'] + 1 |
|  | Apply function to column | df['Age'] = df['Age'].apply(lambda x: x + 1) |
| Merging DataFrames | Merge two DataFrames on common column | df\_merged = pd.merge(df1, df2, on='ID') |
|  | Merge with multiple keys | pd.merge(df1, df2, on=['ID', 'Name']) |
| Concatenating DataFrames | Concatenate along rows (axis=0) | df = pd.concat([df1, df2], axis=0) |
|  | Concatenate along columns (axis=1) | df = pd.concat([df1, df2], axis=1) |
| Removing Data | Drop a column | df.drop('Column', axis=1, inplace=True) |
|  | Drop a row (by index) | df.drop(0, axis=0, inplace=True) |
| Exporting Data | Save to CSV | df.to\_csv('output.csv', index=False) |
|  | Save to Excel | df.to\_excel('output.xlsx', index=False) |
| Indexing & Renaming | Set a new index | df.set\_index('ID', inplace=True) |
|  | Reset index | df.reset\_index(drop=True, inplace=True) |
|  | Rename columns | df.rename(columns={'old\_name': 'new\_name'}, inplace=True) |
| Time Series | Convert column to datetime | df['Date'] = pd.to\_datetime(df['Date']) |
|  | Set datetime column as index | df.set\_index('Date', inplace=True) |
|  | Resample time series data | df.resample('M').mean() |
| Pivoting | Create a pivot table | df.pivot\_table(values='Value', index='Category', aggfunc='sum') |
| Windowing Functions | Rolling window sum (e.g., 3 periods) | df['Rolling\_Sum'] = df['Value'].rolling(3).sum() |
|  | Expanding window sum | df['Expanding\_Sum'] = df['Value'].expanding().sum() |
| String Operations | Convert to lowercase | df['Column'] = df['Column'].str.lower() |
|  | Find substring in column | df['Column'].str.contains('keyword') |
|  | Replace substring | df['Column'] = df['Column'].str.replace('old', 'new') |
| Applying Lambda Functions | Apply lambda function | df['Age'] = df['Age'].apply(lambda x: x + 1) |
| Plotting | Basic Plot (using Matplotlib) | df['Age'].plot() |
|  | Histograms | df['Age'].plot(kind='hist') |
|  | Scatter Plot | df.plot(kind='scatter', x='Age', y='Salary') |
| Multi-Indexing | Create Multi-Index | df.set\_index(['Column1', 'Column2'], inplace=True) |
|  | Access Multi-Index | df.loc['value1', 'value2'] |
| Cross-Tabulation | Cross-tabulate data | pd.crosstab(df['Column1'], df['Column2']) |
| Sampling | Random sampling | df.sample(n=5) |
|  | Sampling with weights | df.sample(n=5, weights='Column', random\_state=42) |
| Categorical Data | Convert to Categorical | df['Category'] = df['Category'].astype('category') |
|  | Get Categories of Categorical Column | df['Category'].cat.categories |
| String Handling (Advanced) | Extract matching pattern | df['Extracted'] = df['Column'].str.extract(r'(\d+)') |
|  | Split strings | df['Split'] = df['Column'].str.split('-') |
| Date/Time Operations | Extract year/month/day from datetime | df['Year'] = df['Date'].dt.year |
|  | Time differences | df['Time\_Diff'] = pd.to\_datetime(df['End\_Date']) - pd.to\_datetime(df['Start\_Date']) |
| Pivot & Unstack | Pivot | df.pivot(index='Date', columns='Category', values='Value') |
|  | Unstack | df.unstack(level=0) |
| Duplication | Find Duplicate Rows | df[df.duplicated()] |
|  | Remove Duplicate Rows | df.drop\_duplicates(inplace=True) |
| Apply Map | Map a function or dictionary | df['Column'] = df['Column'].map(lambda x: x \* 2) |
| Merge/Join with Different Types | Left Join | merged\_df = pd.merge(df1, df2, on='ID', how='left') |
|  | Right Join | merged\_df = pd.merge(df1, df2, on='ID', how='right') |
|  | Outer Join | merged\_df = pd.merge(df1, df2, on='ID', how='outer') |
| Efficient DataFrame Operations | Using `query` method for faster filtering | df.query('Age > 30 and City == "New York"') |
|  | Vectorized Operations | df['New\_Column'] = df['Column1'] \* df['Column2'] |
| DataFrame to Dictionary | Convert DataFrame to Dictionary | df\_dict = df.to\_dict() |
| Plotting with Seaborn/Matplotlib | Seaborn - Pairplot | import seaborn as sns; sns.pairplot(df) |
|  | Seaborn - Heatmap | sns.heatmap(df.corr()) |