**PANDAS COMMANDS**

|  |  |
| --- | --- |
| **COMMAND** | **PURPOSE** |
| **Basic Commands** | |
| import pandas | to import Pandas library |
| pandas.read\_excel(“File\_location”) | To read excel file |
| pandas.read\_csv(“File\_location”) | To read csv file |
| pandas.read\_json(“File\_location”) | To read json file |
| pandas.DataFrame(“<data\_file>”) | To create a data frame from data file |
| **Indexing** **and** **Slicing** | |
| <dataframe\_name>.head() | To display top 5 rows of dataframe |
| <dataframe\_name>.head(3) | To display top 3 rows of dataframe |
| <dataframe\_name>.tail() | To display bottom 5 rows of dataframe |
| <dataframe\_name>.head(3) | To display bottom 3 rows of dataframe |
| <dataframe\_name>.info() | Gives the information of nonnull values |
| <dataframe\_name>.describe() | To display min, max, avg and mean values of all columns |
| <dataframe\_name>.shape | To display number of rows and columns of dataframe |
| <dataframe\_name>[start:stop:step]  <dataframe\_name>[0:9:1] | To display from index 0 to 9 with 1 difference |
| <dataframe\_name>[0:9:2] | To display from index 0 to 9 with 2 difference |
| <dataframe\_name>[“column\_name”] | To display the rows of specified column name |
| <dataframe\_name>[[“column1”, “column2”]] | To display the rows of multiple columns |
| <df\_name>[[“col1”, “col2”]][start:stop:step] | To display the rows of multiple columns with start, stop and step values |
| for <val> in dataframe.iterrows():  print(val) | To print all the rows and columns in tuple format |
| **loc commands (Columns are denoted using names)** | |
| <df\_name>.loc[8] | To display content in row number 8 (includes 8th row) |
| <df\_name>.loc[8,[“col\_name”]] | To display content in row number and col name |
| <df\_name>.loc[0: 20: 1] | To display contents from row 1 to 20 |
| <df\_name>.loc[0:20, “col\_name”] | To display content of only one column from 0 to 20 |
| <df\_name>.loc[0:20, [“col\_name”, ”col\_name”]] | To display content of multiple columns from 0 to 20 |
| <df\_name>.loc[0:20, “col\_name”: ”col\_name”] | To display content of multiple columns between given range from 0 to 20 rows |
| **iloc commands (columns are denoted using index)** | |
| dataset.iloc[0:10] | To print data in between 0 to 9th row (excludes 10th row) |
| dataset.iloc[0,10] | to print data in 0th row of 10th column |
| dataset.iloc[0:10, 1] | To print data in between 0 to 9th row for first column (column) |
| dataset.iloc[0:10, 1:9] | To print data in between 0th to 9th row for first to eighth column |
| dataset.iloc[[1,3,5,6]] | To print data in specified rows |
| dataset.iloc[[1,3,5,6],[2,4,5]] | To print data in specified rows and specified columns |
| dataset.iloc[:,[2,4,5]] | To print all rows of specific columns |
| dataset.iloc[0:20:2,[2,4,5]] | To print data with start stop and skip values of specified rows |
| dataset.iloc[0:20:2,0:10] | To print data for first 20 rows with skip value of 2 and first 10 columns |
| **Sorting dataframe** | |
| df.sort\_values("col\_name") | To print values of column in ascending order |
| df.sort\_values(“col\_name”, ascending=False) | To print values of column in descending order |
| df.sort\_values([“col\_name1”, “col\_name2”]) | Based on first column data will be sorted |
| **Manipulating Dataframe** | |
| **Adding** **Column** | |
| df["Total Marks"] = 0 | To create a new column with value as 0 |
| df["Total Marks"] = df ["English"] + df["Hindi"] + df["Telugu"] + df["Maths"] + df["Science"] + df["EVS"] + df["Sports"] + df["AI"] + df["Sanskrit"] | To create a new column **(Total Marks)**with value as addition of multiple columns |
| **Deleting Column** | |
| df.drop(columns="AI") | To delete the column **AI** just for display purpose |
| df.drop(columns="AI", inplace=True) | To delete the column **AI** from Dataframe permanently |
| **Removing Duplicates** | |
| dataframe.duplicated() | It give Boolean output to show duplicate rows are available |
| dataframe.drop\_duplicates() | It gives output without duplicates, but wont removes duplicate rows from dataframe |
| dataframe.drop\_duplicates(inplace=True) | It removes duplicate rows from dataframe and gives output. Duplicate rows are deleted permanently |
| **Handling Missing Data** | |
| dataframe.fillna(65) | Fills the missing value box with value 65 but for display, not in dataframe |
| dataframe.dropna() | Drops the missing value rows for display but not from dataframe |
| dataframe.dropna(inplace=True) | Drops the missing value rows from dataframe permanently |
|  |  |
| **Data Filtering and Conditional Changes** | |
| dataframe.loc[dataframe["Maths"]>85] | Simple Condition |
| dataframe.loc[(dataframe["Maths"]>70) & (dataframe["Maths"]<98)] | Compound condition |
| dataframe.loc[dataframe["Name"].str.startswith("U")] | Starts with |
| dataframe.loc[dataframe["Name"].str.contains("i")] | Contains |
| dataframe.loc[dataframe["Name"].str.endswith("h")] | Ends with |
| dataframe["Total"] = dataframe["English"] + dataframe["Hindi"] + dataframe["Telugu"] + dataframe["Maths"] + dataframe["Science"] + dataframe["Social"] | Creating a new column and defining values |
| dataframe["Percentage"] = (dataframe["Total"]/600)\*100 | Creating a new column with formula |
| dataframe["Grade"] = "Pass/Fail"  dataframe.loc[dataframe["Percentage"] <= 40,["Grade"]]="Fail"  dataframe.loc[(dataframe["Percentage"] >= 41) & (dataframe["Percentage"] <= 50), ["Grade"]]= "Second Class"  dataframe.loc[dataframe["Percentage"] >= 51, ["Grade"]]= "First Class" | Defining a column and applying logic to select grade based on marks attained |
| **EXPORTING DATAFRAME** | |
| dataframe.to\_excel("C:\<DEST\_PATH>\filename.xlsx” | To save xlsx file with index |
| dataframe.to\_excel("C:\<DEST\_PATH>\filename.xlsx”, index=False | To save xlsx file without index |
| dataframe.to\_json("C:\<DEST\_PATH>\filename.json” | To save json file |