# Pandas

#Importing libraries

1)#python library for numerical and scientific computing. pandas is built on top of numpy

import numpy as np

#importing pandas

import pandas as pd

series = pd.Series(data = [78, 92, 36, 64, 89])

series

|  |
| --- |
| output |
| 0 78  1 92  2 36  3 64  4 89  dtype: int64 |

series.values

|  |
| --- |
| output |
| array([78, 92, 36, 64, 89], dtype=int64) |

series.index

|  |
| --- |
| output |
| RangeIndex(start=0, stop=5, step=1) |

series[1]

|  |
| --- |
| output |
| 92 |

series[1:3]

|  |
| --- |
| output |
| 1 92  2 36  dtype: int64 |

data = pd.Series(data = [700000, 800000, 1600000, 1800000, 30000000], index = ['Swift', 'Jazz', 'Civic', 'Altis', 'Gallardo'])

data

|  |
| --- |
| output |
| Swift 700000  Jazz 800000  Civic 1600000  Altis 1800000  Gallardo 30000000  dtype: int64 |

data['Swift']

|  |
| --- |
| output |
| 700000 |

data['Jazz': 'Gallardo']

|  |
| --- |
| output |
| Jazz 800000  Civic 1600000  Altis 1800000  Gallardo 30000000  dtype: int64 |

2)# create a series out of the dictionary data structure

#Using dictionary to create a series

car\_price\_dict = {'Swift': 700000,

'Jazz' : 800000,

'Civic' : 1600000,

'Altis' : 1800000,

'Gallardo': 30000000

}

car\_price = pd.Series(car\_price\_dict)

car\_price

|  |
| --- |
| output |
| Swift 700000  Jazz 800000  Civic 1600000  Altis 1800000  Gallardo 30000000  dtype: int64 |

#Creating a car price series with a dictionary

car\_price\_dict = {'Swift': 700000,

'Jazz' : 800000,

'Civic' : 1600000,

'Altis' : 1800000,

'Gallardo': 30000000

}

car\_price = pd.Series(car\_price\_dict)

# Creating the car manufacturer series with a dictionary

car\_man\_dict = {'Swift' : 'Maruti',

'Jazz' : 'Honda',

'Civic' : 'Honda',

'Altis' : 'Toyota',

'Gallardo' : 'Lamborghini'}

car\_man = pd.Series(car\_man\_dict)

print(car\_price)

print(car\_man)

|  |
| --- |
| output |
| Swift 700000  Jazz 800000  Civic 1600000  Altis 1800000  Gallardo 30000000  dtype: int64  Swift Maruti  Jazz Honda  Civic Honda  Altis Toyota  Gallardo Lamborghini  dtype: object |

3) # Creating a dataframe

cars = pd.DataFrame({'Price': car\_price , 'Manufacturer' : car\_man})

cars

|  |
| --- |
| output |
| | **Price** | **Manufacturer** | | --- | --- |  |  |  |  | | --- | --- | --- | | **Swift** | 700000 | Maruti | | **Jazz** | 800000 | Honda | | **Civic** | 1600000 | Honda | | **Altis** | 1800000 | Toyota | | **Gallardo** | 30000000 | Lamborghini | |

cars['Price']

|  |
| --- |
| output |
| Swift 700000  Jazz 800000  Civic 1600000  Altis 1800000  Gallardo 30000000  Name: Price, dtype: int64 |

cars['Manufacturer']

|  |
| --- |
| output |
| Swift Maruti  Jazz Honda  Civic Honda  Altis Toyota  Gallardo Lamborghini  Name: Manufacturer, dtype: object |

# 1. From a single series object

#Using dictionary to create a series

car\_price\_dict = {'Swift': 700000,

'Jazz' : 800000,

'Civic' : 1600000,

'Altis' : 1800000,

'Gallardo': 30000000

}

car\_price = pd.Series(car\_price\_dict)

car\_price

#Creating a DataFrame from car\_price Series

pd.DataFrame(car\_price, columns=['CarPrice'])

|  |
| --- |
| Output |
| **CarPrice**   |  |  | | --- | --- | | **Swift** | 700000 | | **Jazz** | 800000 | | **Civic** | 1600000 | | **Altis** | 1800000 | | **Gallardo** | 30000000 | |

# 2. From a list of dictionaries

data = [{'Name': 'Subodh', 'Marks': 28},

{'Name': 'Ram', 'Marks': 27},

{'Name': 'Abdul', 'Marks': 26},

{'Name': 'John', 'Marks': 28}]

pd.DataFrame(data)

|  |
| --- |
| output |
| |  | **Name** | **Marks** | | --- | --- | --- | | **0** | Subodh | 28 | | **1** | Ram | 27 | | **2** | Abdul | 26 | | **3** | John | 28 | |

pd.DataFrame([{'Subodh':20, 'Ram':25,'Abdul':40},

{'Abdul':29, 'John':24}],

index = ['Mathematics', 'Physics'])

|  |
| --- |
| output |
| |  | **Subodh** | **Ram** | **Abdul** | **John** | | --- | --- | --- | --- | --- | | **Mathematics** | 20.0 | 25.0 | 40 | NaN | | **Physics** | NaN | NaN | 29 | 24.0 | |

# 3. From a dictionary of series objects

#Using dictionary to create a series

car\_price\_dict = {'Swift': 700000,

'Jazz' : 800000,

'Civic' : 1600000,

'Altis' : 1800000,

'Gallardo': 30000000

}

car\_price = pd.Series(car\_price\_dict)

car\_man\_dict = {'Swift' : 'Maruti',

'Jazz' : 'Honda',

'Civic' : 'Honda',

'Altis' : 'Toyota',

'Gallardo' : 'Lamborghini'}

car\_man = pd.Series(car\_man\_dict)

cars = pd.DataFrame({'Price': car\_price , 'Manufacturer' : car\_man})

cars

|  |
| --- |
| Output |
| |  | **Price** | **Manufacturer** | | --- | --- | --- | | **Swift** | 700000 | Maruti | | **Jazz** | 800000 | Honda | | **Civic** | 1600000 | Honda | | **Altis** | 1800000 | Toyota | | **Gallardo** | 30000000 | Lamborghini | |

# 4. From an existing file

import pandas as pd

import numpy as np

df = pd.read\_csv('auto\_mpg.csv')

df

|  |
| --- |
| Output |
| |  | **mpg** | **cylinders** | **displacement** | **horsepower** | **weight** | **acceleration** | **model\_year** | **origin** | **name** | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | |  | **0** | 18.0 | 8 | 307.0 | 130.0 | 3504 | 12.0 | 70 | usa | | **1** | 15.0 | 8 | 350.0 | 165.0 | 3693 | 11.5 | 70 | usa | buick skylark 320 | | **2** | 18.0 | 8 | 318.0 | 150.0 | 3436 | 11.0 | 70 | usa | plymouth satellite | | **3** | 16.0 | 8 | 304.0 | 150.0 | 3433 | 12.0 | 70 | usa | amc rebel sst | | **4** | 17.0 | 8 | 302.0 | 140.0 | 3449 | 10.5 | 70 | usa | ford torino | | **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | | **393** | 27.0 | 4 | 140.0 | 86.0 | 2790 | 15.6 | 82 | usa | ford mustang gl | | **394** | 44.0 | 4 | 97.0 | 52.0 | 2130 | 24.6 | 82 | europe | vw pickup | | **395** | 32.0 | 4 | 135.0 | 84.0 | 2295 | 11.6 | 82 | usa | dodge rampage | | **396** | 28.0 | 4 | 120.0 | 79.0 | 2625 | 18.6 | 82 | usa | ford ranger | | **397** | 31.0 | 4 | 119.0 | 82.0 | 2720 | 19.4 | 82 | usa | chevy s-10 |   398 rows × 9 columns |

#5. Head and Tail, describe and info

df.head()

|  |
| --- |
| Output |
| |  | **mpg** | **cylinders** | **displacement** | **Ho rsepower** | **weight** | **acceleration** | **model\_year** | **origin** | **name** | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | **0** | 18.0 | 8 | 307.0 | 130.0 | 3504 | 12.0 | 70 | usa | chevrolet chevelle malibu | | **1** | 15.0 | 8 | 350.0 | 165.0 | 3693 | 11.5 | 70 | usa | buick skylark 320 | | **2** | 18.0 | 8 | 318.0 | 150.0 | 3436 | 11.0 | 70 | usa | plymouth satellite | | **3** | 16.0 | 8 | 304.0 | 150.0 | 3433 | 12.0 | 70 | usa | amc rebel sst | | **4** | 17.0 | 8 | 302.0 | 140.0 | 3449 |  |  |  |  | |

df.tail()

|  |
| --- |
| Output |
| |  | **mpg** | **cylinders** | **displacement** | **horsepower** | **weight** | **acceleration** | **model\_year** | **origin** | **name** | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | **393** | 27.0 | 4 | 140.0 | 86.0 | 2790 | 15.6 | 82 | usa | ford mustang gl | | **394** | 44.0 | 4 | 97.0 | 52.0 | 2130 | 24.6 | 82 | europe | vw pickup | | **395** | 32.0 | 4 | 135.0 | 84.0 | 2295 | 11.6 | 82 | usa | dodge rampage | | **396** | 28.0 | 4 | 120.0 | 79.0 | 2625 | 18.6 | 82 | usa | ford ranger | | **397** | 31.0 | 4 | 119.0 | 82.0 | 2720 | 19.4 | 82 | usa | chevy s-10 | |

df.describe()

|  |
| --- |
| Output |
| |  | **mpg** | **cylinders** | **displacement** | **horsepower** | **weight** | **acceleration** | **model\_year** | | --- | --- | --- | --- | --- | --- | --- | --- | | **count** | 398.000000 | 398.000000 | 398.000000 | 392.000000 | 398.000000 | 398.000000 | 398.000000 | | **mean** | 23.514573 | 5.454774 | 193.425879 | 104.469388 | 2970.424623 | 15.568090 | 76.010050 | | **std** | 7.815984 | 1.701004 | 104.269838 | 38.491160 | 846.841774 | 2.757689 | 3.697627 | | **min** | 9.000000 | 3.000000 | 68.000000 | 46.000000 | 1613.000000 | 8.000000 | 70.000000 | | **25%** | 17.500000 | 4.000000 | 104.250000 | 75.000000 | 2223.750000 | 13.825000 | 73.000000 | | **50%** | 23.000000 | 4.000000 | 148.500000 | 93.500000 | 2803.500000 | 15.500000 | 76.000000 | | **75%** | 29.000000 | 8.000000 | 262.000000 | 126.000000 | 3608.000000 | 17.175000 | 79.000000 | | **max** | 46.600000 | 8.000000 | 455.000000 | 230.000000 | 5140.000000 | 24.800000 | 82.00000 | |

df.info()

|  |
| --- |
| Output |
| <class 'pandas.core.frame.DataFrame'>  RangeIndex: 398 entries, 0 to 397  Data columns (total 9 columns):  # Column Non-Null Count Dtype  --- ------ -------------- -----  0 mpg 398 non-null float64  1 cylinders 398 non-null int64  2 displacement 398 non-null float64  3 horsepower 392 non-null float64  4 weight 398 non-null int64  5 acceleration 398 non-null float64  6 model\_year 398 non-null int64  7 origin 398 non-null object  8 name 398 non-null object  dtypes: float64(4), int64(3), object(2)  memory usage: 28.1+ KB |

# 6. Dropping null values

df.dropna(inplace = True)

df.info()

|  |
| --- |
| Output |
| <class 'pandas.core.frame.DataFrame'>  Int64Index: 392 entries, 0 to 397  Data columns (total 9 columns):  # Column Non-Null Count Dtype  --- ------ -------------- -----  0 mpg 392 non-null float64  1 cylinders 392 non-null int64  2 displacement 392 non-null float64  3 horsepower 392 non-null float64  4 weight 392 non-null int64  5 acceleration 392 non-null float64  6 model\_year 392 non-null int64  7 origin 392 non-null object  8 name 392 non-null object  dtypes: float64(4), int64(3), object(2)  memory usage: 30.6+ KB |

#7. Selecting a subset of the data

df['name']

|  |
| --- |
| Output |
| 0 chevrolet chevelle malibu  1 buick skylark 320  2 plymouth satellite  3 amc rebel sst  4 ford torino  ...  393 ford mustang gl  394 vw pickup  395 dodge rampage  396 ford ranger  397 chevy s-10  Name: name, Length: 392, dtype: object |

df[['name']]

|  |
| --- |
| Output |
| |  | **name** | | --- | --- | | **0** | chevrolet chevelle malibu | | **1** | buick skylark 320 | | **2** | plymouth satellite | | **3** | amc rebel sst | | **4** | ford torino | | **...** | ... | | **393** | ford mustang gl | | **394** | vw pickup | | **395** | dodge rampage | | **396** | ford ranger | | **397** | chevy s-10 |   392 rows × 1 columns |

df[['name', 'origin', 'model\_year', 'mpg']]

|  |
| --- |
| Output |
| |  | **name** | **origin** | **model\_year** | **mpg** | | --- | --- | --- | --- | --- | | **0** | chevrolet chevelle malibu | usa | 70 | 18.0 | | **1** | buick skylark 320 | usa | 70 | 15.0 | | **2** | plymouth satellite | usa | 70 | 18.0 | | **3** | amc rebel sst | usa | 70 | 16.0 | | **4** | ford torino | usa | 70 | 17.0 | | **...** | ... | ... | ... | ... | | **393** | ford mustang gl | usa | 82 | 27.0 | | **394** | vw pickup | europe | 82 | 44.0 | | **395** | dodge rampage | usa | 82 | 32.0 | | **396** | ford ranger | usa | 82 | 28.0 | | **397** | chevy s-10 | usa | 82 | 31.0 |   392 rows × 4 columns |

#8. Setting custom index:

#creating a subset using head

df\_head = df.head()

#Setting name as custom index

df\_head.set\_index('name', inplace = True)

df\_head

|  |
| --- |
| 0utput |
| |  | **mpg** | **cylinders** | **displacement** | **horsepower** | **weight** | **acceleration** | **model\_year** | **origin** | | --- | --- | --- | --- | --- | --- | --- | --- | --- | | **name** |  |  |  |  |  |  |  |  | | **chevrolet chevelle malibu** | 18.0 | 8 | 307.0 | 130.0 | 3504 | 12.0 | 70 | usa | | **buick skylark 320** | 15.0 | 8 | 350.0 | 165.0 | 3693 | 11.5 | 70 | usa | | **plymouth satellite** | 18.0 | 8 | 318.0 | 150.0 | 3436 | 11.0 | 70 | usa | | **amc rebel sst** | 16.0 | 8 | 304.0 | 150.0 | 3433 | 12.0 | 70 | usa | | **ford torino** | 17.0 | 8 | 302.0 | 140.0 | 3449 | 10.5 | 70 | usa | |

df = pd.read\_csv('auto\_mpg.csv')

df

|  |
| --- |
| Output |
| |  | **mpg** | **cylinders** | **displacement** | **horsepower** | **weight** | **acceleration** | **model\_year** | **origin** | **name** | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | **0** | 18.0 | 8 | 307.0 | 130.0 | 3504 | 12.0 | 70 | usa | chevrolet chevelle malibu | | **1** | 15.0 | 8 | 350.0 | 165.0 | 3693 | 11.5 | 70 | usa | buick skylark 320 | | **2** | 18.0 | 8 | 318.0 | 150.0 | 3436 | 11.0 | 70 | usa | plymouth satellite | | **3** | 16.0 | 8 | 304.0 | 150.0 | 3433 | 12.0 | 70 | usa | amc rebel sst | | **4** | 17.0 | 8 | 302.0 | 140.0 | 3449 | 10.5 | 70 | usa | ford torino | | **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | | **393** | 27.0 | 4 | 140.0 | 86.0 | 2790 | 15.6 | 82 | usa | ford mustang gl | | **394** | 44.0 | 4 | 97.0 | 52.0 | 2130 | 24.6 | 82 | europe | vw pickup | | **395** | 32.0 | 4 | 135.0 | 84.0 | 2295 | 11.6 | 82 | usa | dodge rampage | | **396** | 28.0 | 4 | 120.0 | 79.0 | 2625 | 18.6 | 82 | usa | ford ranger | | **397** | 31.0 | 4 | 119.0 | 82.0 | 2720 | 19.4 | 82 | usa | chevy s-10 |   398 rows × 9 columns |

df.iloc[2,1]

|  |
| --- |
| output |
| 8 |

df.iloc[2,-1]

|  |
| --- |
| output |
| 'plymouth satellite' |

df.iloc[1:5, 4:6]

|  |
| --- |
| output |
| |  | **weight** | **acceleration** | | --- | --- | --- | | **1** | 3693 | 11.5 | | **2** | 3436 | 11.0 | | **3** | 3433 | 12.0 | | **4** | 3449 | 10.5 | |

9. #creating a subset using head. 'df' refers to XYZ Custom Cars DataFrame.

df\_head = df.head()

#Setting name as custom index

df\_head.set\_index('name', inplace = True)

df\_head.loc['buick skylark 320': 'amc rebel sst']

|  |
| --- |
| Output |
| |  | **mpg** | **cylinders** | **displacement** | **horsepower** | **weight** | **acceleration** | **model\_year** | **origin** | | --- | --- | --- | --- | --- | --- | --- | --- | --- | | **name** |  |  |  |  |  |  |  |  | | **buick skylark 320** | 15.0 | 8 | 350.0 | 165.0 | 3693 | 11.5 | 70 | usa | | **plymouth satellite** | 18.0 | 8 | 318.0 | 150.0 | 3436 | 11.0 | 70 | usa | | **amc rebel sst** | 16.0 | 8 | 304.0 | 150.0 | 3433 | 12.0 | 70 | usa | |

#Subsetting from the full dataset

df.loc[0:5, ['cylinders', 'horsepower', 'name']]

|  |
| --- |
| Output |
| |  | **cylinders** | **horsepower** | **name** | | --- | --- | --- | --- | | **0** | 8 | 130.0 | chevrolet chevelle malibu | | **1** | 8 | 165.0 | buick skylark 320 | | **2** | 8 | 150.0 | plymouth satellite | | **3** | 8 | 150.0 | amc rebel sst | | **4** | 8 | 140.0 | ford torino | | **5** | 8 | 198.0 | ford galaxie 500 | |

# 10. Fuel efficient - Cars designed with low power and high fuel efficiency - MPG > 29, Horsepower < 93.5, Weight < 2500

# 11. Muscle Cars - Intermediate sized cars designed for high performance - Displacement >262, Horsepower > 126, Weight in range[2800, 3600]

# 12. SUV- Big sized cars designed for high performance, long distance trips and family comfort - Horsepower > 140 , Weight > 4500

# 13. Racecar- Cars specifically designed for race tracks , Weight <2223, acceleration > 17

# Fuel efficient

# MPG > 29, Horsepower < 93.5,

# Weight < 2500

df.loc[(df['mpg'] > 29) & (df['horsepower'] < 93.5) & (df['weight'] < 2500)]

|  |
| --- |
| Output |
| |  | **mpg** | **cylinders** | **displacement** | **horsepower** | **weight** | **acceleration** | **model\_year** | **origin** | **name** | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | **51** | 30.0 | 4 | 79.0 | 70.0 | 2074 | 19.5 | 71 | europe | peugeot 304 | | **52** | 30.0 | 4 | 88.0 | 76.0 | 2065 | 14.5 | 71 | europe | fiat 124b | | **53** | 31.0 | 4 | 71.0 | 65.0 | 1773 | 19.0 | 71 | japan | toyota corolla 1200 | | **54** | 35.0 | 4 | 72.0 | 69.0 | 1613 | 18.0 | 71 | japan | datsun 1200 | | **129** | 31.0 | 4 | 79.0 | 67.0 | 1950 | 19.0 | 74 | japan | datsun b210 | | **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | | **384** | 32.0 | 4 | 91.0 | 67.0 | 1965 | 15.7 | 82 | japan | honda civic (auto) | | **385** | 38.0 | 4 | 91.0 | 67.0 | 1995 | 16.2 | 82 | japan | datsun 310 gx | | **391** | 36.0 | 4 | 135.0 | 84.0 | 2370 | 13.0 | 82 | usa | dodge charger 2.2 | | **394** | 44.0 | 4 | 97.0 | 52.0 | 2130 | 24.6 | 82 | europe | vw pickup | | **395** | 32.0 | 4 | 135.0 | 84.0 | 2295 | 11.6 | 82 | usa | dodge rampage |   81 rows × 9 columns |

# Muscle cars

# Displacement >262, Horsepower > 126, Weight in range[2800, 3600]

df.loc[(df['displacement'] > 262) & (df['horsepower'] > 126) & (df['weight'] >=2800) & (df['weight'] <= 3600)]

|  |
| --- |
| output |
| |  | **mpg** | **cylinders** | **displacement** | **horsepower** | **weight** | **acceleration** | **model\_year** | **origin** | **name** | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | **0** | 18.0 | 8 | 307.0 | 130.0 | 3504 | 12.0 | 70 | usa | chevrolet chevelle malibu | | **2** | 18.0 | 8 | 318.0 | 150.0 | 3436 | 11.0 | 70 | usa | plymouth satellite | | **3** | 16.0 | 8 | 304.0 | 150.0 | 3433 | 12.0 | 70 | usa | amc rebel sst | | **4** | 17.0 | 8 | 302.0 | 140.0 | 3449 | 10.5 | 70 | usa | ford torino | | **10** | 15.0 | 8 | 383.0 | 170.0 | 3563 | 10.0 | 70 | usa | dodge challenger se | | **13** | 14.0 | 8 | 455.0 | 225.0 | 3086 | 10.0 | 70 | usa | buick estate wagon (sw) | | **121** | 15.0 | 8 | 318.0 | 150.0 | 3399 | 11.0 | 73 | usa | dodge dart custom | | **166** | 13.0 | 8 | 302.0 | 129.0 | 3169 | 12.0 | 75 | usa | ford mustang ii | | **251** | 20.2 | 8 | 302.0 | 139.0 | 3570 | 12.8 | 78 | usa | mercury monarch ghia | | **262** | 19.2 | 8 | 305.0 | 145.0 | 3425 | 13.2 | 78 | usa | chevrolet monte carlo landau | | **264** | 18.1 | 8 | 302.0 | 139.0 | 3205 | 11.2 | 78 | usa | ford futura | |

# SUV

# Horsepower > 140 , Weight > 4500

df.loc[(df['horsepower'] > 140) & (df['weight'] >=4500)]

|  |
| --- |
| output |
| |  | **mpg** | **cylinders** | **displacement** | **horsepower** | **weight** | **acceleration** | **model\_year** | **origin** | **name** | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | **25** | 10.0 | 8 | 360.0 | 215.0 | 4615 | 14.0 | 70 | usa | ford f250 | | **28** | 9.0 | 8 | 304.0 | 193.0 | 4732 | 18.5 | 70 | usa | hi 1200d | | **42** | 12.0 | 8 | 383.0 | 180.0 | 4955 | 11.5 | 71 | usa | dodge monaco (sw) | | **43** | 13.0 | 8 | 400.0 | 170.0 | 4746 | 12.0 | 71 | usa | ford country squire (sw) | | **44** | 13.0 | 8 | 400.0 | 175.0 | 5140 | 12.0 | 71 | usa | pontiac safari (sw) | | **67** | 11.0 | 8 | 429.0 | 208.0 | 4633 | 11.0 | 72 | usa | mercury marquis | | **68** | 13.0 | 8 | 350.0 | 155.0 | 4502 | 13.5 | 72 | usa | buick lesabre custom | | **90** | 12.0 | 8 | 429.0 | 198.0 | 4952 | 11.5 | 73 | usa | mercury marquis brougham | | **94** | 13.0 | 8 | 440.0 | 215.0 | 4735 | 11.0 | 73 | usa | chrysler new yorker brougham | | **95** | 12.0 | 8 | 455.0 | 225.0 | 4951 | 11.0 | 73 | usa | buick electra 225 custom | | **103** | 11.0 | 8 | 400.0 | 150.0 | 4997 | 14.0 | 73 | usa | chevrolet impala | | **104** | 12.0 | 8 | 400.0 | 167.0 | 4906 | 12.5 | 73 | usa | ford country | | **105** | 13.0 | 8 | 360.0 | 170.0 | 4654 | 13.0 | 73 | usa | plymouth custom suburb | | **137** | 13.0 | 8 | 350.0 | 150.0 | 4699 | 14.5 | 74 | usa | buick century luxus (sw) | | **156** | 16.0 | 8 | 400.0 | 170.0 | 4668 | 11.5 | 75 | usa | pontiac catalina | | **159** | 14.0 | 8 | 351.0 | 148.0 | 4657 | 13.5 | 75 | usa | ford ltd | |

# Racecar

# Weight <2223, acceleration > 17

df.loc[(df['acceleration'] > 17) & (df['weight'] < 2223)]

|  |
| --- |
| output |
| | **mpg** | **cylinders** | **displacement** | **horsepower** | **weight** | **acceleration** | **model\_year** | **origin** | **name** | | --- | --- | --- | --- | --- | --- | --- | --- | --- | | **19** | 26.0 | 4 | 97.0 | 46.0 | 1835 | 20.5 | 70 | europe | volkswagen 1131 deluxe sedan | | **32** | 25.0 | 4 | 98.0 | NaN | 2046 | 19.0 | 71 | usa | ford pinto | | **51** | 30.0 | 4 | 79.0 | 70.0 | 2074 | 19.5 | 71 | europe | peugeot 304 | | **53** | 31.0 | 4 | 71.0 | 65.0 | 1773 | 19.0 | 71 | japan | toyota corolla 1200 | | **54** | 35.0 | 4 | 72.0 | 69.0 | 1613 | 18.0 | 71 | japan | datsun 1200 | | **55** | 27.0 | 4 | 97.0 | 60.0 | 1834 | 19.0 | 71 | europe | volkswagen model 111 | | **56** | 26.0 | 4 | 91.0 | 70.0 | 1955 | 20.5 | 71 | usa | plymouth cricket | | **79** | 26.0 | 4 | 96.0 | 69.0 | 2189 | 18.0 | 72 | europe | renault 12 (sw) | | **102** | 26.0 | 4 | 97.0 | 46.0 | 1950 | 21.0 | 73 | europe | volkswagen super beetle | | **117** | 29.0 | 4 | 68.0 | 49.0 | 1867 | 19.5 | 73 | europe | fiat 128 | | **129** | 31.0 | 4 | 79.0 | 67.0 | 1950 | 19.0 | 74 | japan | datsun b210 | | **131** | 32.0 | 4 | 71.0 | 65.0 | 1836 | 21.0 | 74 | japan | toyota corolla 1200 | | **145** | 32.0 | 4 | 83.0 | 61.0 | 2003 | 19.0 | 74 | japan | datsun 710 | | **181** | 33.0 | 4 | 91.0 | 53.0 | 1795 | 17.5 | 75 | japan | honda civic cvcc | | **195** | 29.0 | 4 | 85.0 | 52.0 | 2035 | 22.2 | 76 | usa | chevrolet chevette | | **196** | 24.5 | 4 | 98.0 | 60.0 | 2164 | 22.1 | 76 | usa | chevrolet woody | | **198** | 33.0 | 4 | 91.0 | 53.0 | 1795 | 17.4 | 76 | japan | honda civic | | **216** | 31.5 | 4 | 98.0 | 68.0 | 2045 | 18.5 | 77 | japan | honda accord cvcc | | **218** | 36.0 | 4 | 79.0 | 58.0 | 1825 | 18.6 | 77 | europe | renault 5 gtl | | **244** | 43.1 | 4 | 90.0 | 48.0 | 1985 | 21.5 | 78 | europe | volkswagen rabbit custom diesel | | **246** | 32.8 | 4 | 78.0 | 52.0 | 1985 | 19.4 | 78 | japan | mazda glc deluxe | | **247** | 39.4 | 4 | 85.0 | 70.0 | 2070 | 18.6 | 78 | japan | datsun b210 gx | | **303** | 31.8 | 4 | 85.0 | 65.0 | 2020 | 19.2 | 79 | japan | datsun 210 | | **310** | 38.1 | 4 | 89.0 | 60.0 | 1968 | 18.8 | 80 | japan | toyota corolla tercel | | **322** | 46.6 | 4 | 86.0 | 65.0 | 2110 | 17.9 | 80 | japan | mazda glc | | **324** | 40.8 | 4 | 85.0 | 65.0 | 2110 | 19.2 | 80 | japan | datsun 210 | | **325** | 44.3 | 4 | 90.0 | 48.0 | 2085 | 21.7 | 80 | europe | vw rabbit c (diesel) | | **330** | 40.9 | 4 | 85.0 | NaN | 1835 | 17.3 | 80 | europe | renault lecar deluxe | | **331** | 33.8 | 4 | 97.0 | 67.0 | 2145 | 18.0 | 80 | japan | subaru dl | | **346** | 32.3 | 4 | 97.0 | 67.0 | 2065 | 17.8 | 81 | japan | subaru | | **347** | 37.0 | 4 | 85.0 | 65.0 | 1975 | 19.4 | 81 | japan | datsun 210 mpg | | **348** | 37.7 | 4 | 89.0 | 62.0 | 2050 | 17.3 | 81 | japan | toyota tercel | | **376** | 37.0 | 4 | 91.0 | 68.0 | 2025 | 18.2 | 82 | japan | mazda glc custom l | | **377** | 31.0 | 4 | 91.0 | 68.0 | 1970 | 17.6 | 82 | japan | mazda glc custom | | **379** | 36.0 | 4 | 98.0 | 70.0 | 2125 | 17.3 | 82 | usa | mercury lynx l | | **394** | 44.0 | 4 | 97.0 | 52.0 | 2130 | 24.6 | 82 | europe | vw pickup | |

#14. adding new column

marks = {'Chemistry': [67,90,66,32],

'Physics': [45,92,72,40],

'Mathematics': [50,87,81,12],

'English': [19,90,72,68]}

marks\_df = pd.DataFrame(marks, index = ['Subodh', 'Ram', 'Abdul', 'John'])

marks\_df

|  |
| --- |
| output |
| |  | **Chemistry** | **Physics** | **Mathematics** | **English** | | --- | --- | --- | --- | --- | | **Subodh** | 67 | 45 | 50 | 19 | | **Ram** | 90 | 92 | 87 | 90 | | **Abdul** | 66 | 72 | 81 | 72 | | **John** | 32 | 40 | 12 | 68 | |

marks\_df['Total'] = marks\_df['Chemistry'] + marks\_df['Physics'] + marks\_df['Mathematics'] + marks\_df['English']

marks\_df

|  |
| --- |
| output |
| |  | **Chemistry** | **Physics** | **Mathematics** | **English** | **Total** | | --- | --- | --- | --- | --- | --- | | **Subodh** | 67 | 45 | 50 | 19 | 181 | | **Ram** | 90 | 92 | 87 | 90 | 359 | | **Abdul** | 66 | 72 | 81 | 72 | 291 | | **John** | 32 | 40 | 12 | 68 | 152 | |

marks\_df.drop(columns = 'Total', inplace = True)

marks\_df

|  |
| --- |
| Output |
| |  | **Chemistry** | **Physics** | **Mathematics** | **English** | | --- | --- | --- | --- | --- | | **Subodh** | 67 | 45 | 50 | 19 | | **Ram** | 90 | 92 | 87 | 90 | | **Abdul** | 66 | 72 | 81 | 72 | | **John** | 32 | 40 | 12 | 68 | |

**# mask values**

marks = [{'Chemistry': 67, 'Physics': 45, 'Mathematics': 50, 'English' : 19},

{'Chemistry': 90, 'Physics': 92, 'Mathematics': 87, 'English' : 90},

{'Chemistry': 66, 'Physics': 72, 'Mathematics': 81, 'English' : 72},

{'Chemistry': 32, 'Physics': 40, 'Mathematics': 12, 'English' : 68}]

marks\_df = pd.DataFrame(marks, index = ['Subodh', 'Ram', 'Abdul', 'John'])

marks\_df

|  |
| --- |
| output |
| |  | **Chemistry** | **Physics** | **Mathematics** | **English** | | --- | --- | --- | --- | --- | | **Subodh** | 67 | 45 | 50 | 19 | | **Ram** | 90 | 92 | 87 | 90 | | **Abdul** | 66 | 72 | 81 | 72 | | **John** | 32 | 40 | 12 | 68 | |

f = marks\_df < 33

marks\_df.mask(f, 'Fail')

|  |
| --- |
| output |
| | **Chemistry** | **Physics** | **Mathematics** | **English** | | --- | --- | --- | --- | | **Subodh** | 67 | 45 | 50 | Fail | | **Ram** | 90 | 92 | 87 | 90 | | **Abdul** | 66 | 72 | 81 | 72 | | **John** | Fail | 40 | Fail | 68 | |

#XYZ Custom cars want the data sorted according to the number of cylinders.

df.sort\_values(by = 'cylinders')

|  |
| --- |
| Output |
| | **mpg** | **cylinders** | **displacement** | **horsepower** | **weight** | **acceleration** | **model\_year** | **origin** | **name** | | --- | --- | --- | --- | --- | --- | --- | --- | --- | | **111** | 18.0 | 3 | 70.0 | 90.0 | 2124 | 13.5 | 73 | japan | maxda rx3 | | **71** | 19.0 | 3 | 70.0 | 97.0 | 2330 | 13.5 | 72 | japan | mazda rx2 coupe | | **334** | 23.7 | 3 | 70.0 | 100.0 | 2420 | 12.5 | 80 | japan | mazda rx-7 gs | | **243** | 21.5 | 3 | 80.0 | 110.0 | 2720 | 13.5 | 77 | japan | mazda rx-4 | | **267** | 27.5 | 4 | 134.0 | 95.0 | 2560 | 14.2 | 78 | japan | toyota corona | | **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | | **86** | 14.0 | 8 | 304.0 | 150.0 | 3672 | 11.5 | 73 | usa | amc matador | | **285** | 17.0 | 8 | 305.0 | 130.0 | 3840 | 15.4 | 79 | usa | chevrolet caprice classic | | **286** | 17.6 | 8 | 302.0 | 129.0 | 3725 | 13.4 | 79 | usa | ford ltd landau | | **92** | 13.0 | 8 | 351.0 | 158.0 | 4363 | 13.0 | 73 | usa | ford ltd | | **0** | 18.0 | 8 | 307.0 | 130.0 | 3504 | 12.0 | 70 | usa | chevrolet chevelle malibu |   398 rows × 9 columns |

#There is a requirement in which the cars that have lowest acceleration must be assessed. It is also to be checked that which cars have higher horsepower despite having lower acceleration.

df.sort\_values(['acceleration', 'horsepower'], ascending = (1,0))

|  |
| --- |
| Output |
| | **mpg** | **cylinders** | **displacement** | **horsepower** | **weight** | **acceleration** | **model\_year** | **origin** | **name** | | --- | --- | --- | --- | --- | --- | --- | --- | --- | | **11** | 14.0 | 8 | 340.0 | 160.0 | 3609 | 8.0 | 70 | usa | plymouth 'cuda 340 | | **7** | 14.0 | 8 | 440.0 | 215.0 | 4312 | 8.5 | 70 | usa | plymouth fury iii | | **9** | 15.0 | 8 | 390.0 | 190.0 | 3850 | 8.5 | 70 | usa | amc ambassador dpl | | **6** | 14.0 | 8 | 454.0 | 220.0 | 4354 | 9.0 | 70 | usa | chevrolet impala | | **116** | 16.0 | 8 | 400.0 | 230.0 | 4278 | 9.5 | 73 | usa | pontiac grand prix | | **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | | **195** | 29.0 | 4 | 85.0 | 52.0 | 2035 | 22.2 | 76 | usa | chevrolet chevette | | **59** | 23.0 | 4 | 97.0 | 54.0 | 2254 | 23.5 | 72 | europe | volkswagen type 3 | | **326** | 43.4 | 4 | 90.0 | 48.0 | 2335 | 23.7 | 80 | europe | vw dasher (diesel) | | **394** | 44.0 | 4 | 97.0 | 52.0 | 2130 | 24.6 | 82 | europe | vw pickup | | **299** | 27.2 | 4 | 141.0 | 71.0 | 3190 | 24.8 | 79 | europe | peugeot 504 |   398 rows × 9 columns |

#the board of XYZ custom cars wants to know about minimum and maximum of all the numerical columns

#Using list comprehension to get the numerical columns

list1 = [col for col in df.columns if df[col].dtype in ['float', 'int64']]

df[list1].agg(['min', 'max'])

|  |
| --- |
| Output |
| |  | **mpg** | **cylinders** | **displacement** | **horsepower** | **weight** | **acceleration** | **model\_year** | | --- | --- | --- | --- | --- | --- | --- | --- | | **min** | 9.0 | 3 | 68.0 | 46.0 | 1613 | 8.0 | 70 | | **max** | 46.6 | 8 | 455.0 | 230.0 | 5140 | 24.8 | 82 | |

#XYZ custom cars want to know the number of cars manufactured in each year.

df.groupby(['model\_year']).count()[['name']]

|  |
| --- |
| Output |
| |  | **name** | | --- | --- | | **model\_year** |  | | **70** | 29 | | **71** | 28 | | **72** | 28 | | **73** | 40 | | **74** | 27 | | **75** | 30 | | **76** | 34 | | **77** | 28 | | **78** | 36 | | **79** | 29 | | **80** | 29 | | **81** | 29 | | **82** | 31 | |

#Some senior engineers in XYZ custom cars want to understand about the effect of model year and number of cylinders on horsepower.

#Creating a DataFrame grouped on cylinders and model\_year and finding mean, min and max of horsepower

grouped\_multiple = df.groupby(['cylinders', 'model\_year']).agg({'horsepower': ['mean', 'min', 'max']})

#Naming columns in grouped DataFrame

grouped\_multiple.columns = ['hp\_mean', 'hp\_min', 'hp\_max']

#Resetting index

grouped\_multiple = grouped\_multiple.reset\_index()

#Viewing head of resulting DataFrame

grouped\_multiple.head()

|  |
| --- |
| Output |
| | **cylinders** | **model\_year** | **hp\_mean** | **hp\_min** | **hp\_max** | | --- | --- | --- | --- | --- | | **0** | 3 | 72 | 97.000000 | 97.0 | 97.0 | | **1** | 3 | 73 | 90.000000 | 90.0 | 90.0 | | **2** | 3 | 77 | 110.000000 | 110.0 | 110.0 | | **3** | 3 | 80 | 100.000000 | 100.0 | 100.0 | | **4** | 4 | 70 | 87.714286 | 46.0 | 113.0 | |

#he engineers at XYZ Custom Cars want to know about the relationship between model year and acceleration of cars.

df.groupby(['model\_year']).mean()[['acceleration']]

|  |
| --- |
| Output |
| |  | **acceleration** | | --- | --- | | **model\_year** |  | | **70** | 12.948276 | | **71** | 15.142857 | | **72** | 15.125000 | | **73** | 14.312500 | | **74** | 16.203704 | | **75** | 16.050000 | | **76** | 15.941176 | | **77** | 15.435714 | | **78** | 15.805556 | | **79** | 15.813793 | | **80** | 16.934483 | | **81** | 16.306897 | | **82** | 16.638710 | |

#The engineers at XYZ Custom Cars want to know the frequency distribution of different number of cylinders across different years.

pd.crosstab(df['model\_year'], df['cylinders'])

|  |
| --- |
| Output |
| | **cylinders** | **3** | **4** | **5** | **6** | **8** | | --- | --- | --- | --- | --- | --- | | **model\_year** |  |  |  |  |  | | **70** | 0 | 7 | 0 | 4 | 18 | | **71** | 0 | 13 | 0 | 8 | 7 | | **72** | 1 | 14 | 0 | 0 | 13 | | **73** | 1 | 11 | 0 | 8 | 20 | | **74** | 0 | 15 | 0 | 7 | 5 | | **75** | 0 | 12 | 0 | 12 | 6 | | **76** | 0 | 15 | 0 | 10 | 9 | | **77** | 1 | 14 | 0 | 5 | 8 | | **78** | 0 | 17 | 1 | 12 | 6 | | **79** | 0 | 12 | 1 | 6 | 10 | | **80** | 1 | 25 | 1 | 2 | 0 | | **81** | 0 | 21 | 0 | 7 | 1 | | **82** | 0 | 28 | 0 | 3 | 0 | |

#The engineers at XYZ custom cars want to know the mean of all the numerical attributes of cars for each year

pivot1 = pd.pivot\_table(df, index = 'model\_year', aggfunc=np.mean)

pivot1

|  |
| --- |
| Output |
| | **acceleration** | **cylinders** | **displacement** | **horsepower** | **mpg** | **weight** | | --- | --- | --- | --- | --- | --- | | **model\_year** |  |  |  |  |  |  | | **70** | 12.948276 | 6.758621 | 281.413793 | 147.827586 | 17.689655 | 3372.793103 | | **71** | 15.142857 | 5.571429 | 209.750000 | 107.037037 | 21.250000 | 2995.428571 | | **72** | 15.125000 | 5.821429 | 218.375000 | 120.178571 | 18.714286 | 3237.714286 | | **73** | 14.312500 | 6.375000 | 256.875000 | 130.475000 | 17.100000 | 3419.025000 | | **74** | 16.203704 | 5.259259 | 171.740741 | 94.230769 | 22.703704 | 2877.925926 | | **75** | 16.050000 | 5.600000 | 205.533333 | 101.066667 | 20.266667 | 3176.800000 | | **76** | 15.941176 | 5.647059 | 197.794118 | 101.117647 | 21.573529 | 3078.735294 | | **77** | 15.435714 | 5.464286 | 191.392857 | 105.071429 | 23.375000 | 2997.357143 | | **78** | 15.805556 | 5.361111 | 177.805556 | 99.694444 | 24.061111 | 2861.805556 | | **79** | 15.813793 | 5.827586 | 206.689655 | 101.206897 | 25.093103 | 3055.344828 | | **80** | 16.934483 | 4.137931 | 115.827586 | 77.481481 | 33.696552 | 2436.655172 | | **81** | 16.306897 | 4.620690 | 135.310345 | 81.035714 | 30.334483 | 2522.931034 | | **82** | 16.638710 | 4.193548 | 128.870968 | 81.466667 | 31.709677 | 2453.548387 | |

**# Panda plots**

#A scatter plot to visualize the trend of acceleration in different years.

df.plot(x = 'model\_year', y = 'acceleration', marker = 'o', kind = 'scatter');

|  |
| --- |
| Output |
|  |

# A bar plot to visualize mean acceleration in different years.

df.groupby('model\_year').mean()[['acceleration']].plot(kind = 'bar');

|  |
| --- |
| Output |
|  |

#A histogram to visualize the frequency distribution of cylinders

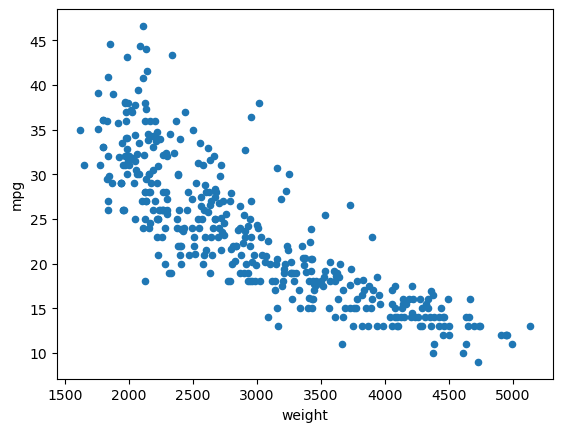
df['cylinders'].plot(kind = 'hist')

|  |
| --- |
| Output  <AxesSubplot:ylabel='Frequency'> |
|  |

#A scatter plot to visualize the relationship between weight and mpg.

df.plot(x = 'weight', y = 'mpg', kind = 'scatter')

|  |
| --- |
| Output |
| <AxesSubplot:xlabel='weight', ylabel='mpg'> |



#A bar plot to visualize the sorted mean values of acceleration with respect to number of cylinders.

df.groupby('cylinders').mean().sort\_values('acceleration')[['acceleration']].plot(kind = 'bar')

|  |
| --- |
| Output |
| <AxesSubplot:xlabel='cylinders'> |

# index preservation

marks = {'Chemistry': [67,90,66,32],

'Physics': [45,92,72,40],

'Mathematics': [50,87,81,12],

'English': [19,90,72,68]}

marks\_df = pd.DataFrame(marks, index = ['Subodh', 'Ram', 'Abdul', 'John'])

marks\_df

|  |
| --- |
| Output |
| |  | **Chemistry** | **Physics** | **Mathematics** | **English** | | --- | --- | --- | --- | --- | | **Subodh** | 67 | 45 | 50 | 19 | | **Ram** | 90 | 92 | 87 | 90 | | **Abdul** | 66 | 72 | 81 | 72 | | **John** | 32 | 40 | 12 | 68 | |

#encrypting marks as sine of marks

encrypted\_marks = np.sin(marks\_df)

encrypted\_marks

|  |
| --- |
| Output |
| | **Chemistry** | **Physics** | **Mathematics** | **English** | | --- | --- | --- | --- | | **Subodh** | -0.855520 | 0.850904 | -0.262375 | 0.149877 | | **Ram** | 0.893997 | -0.779466 | -0.821818 | 0.893997 | | **Abdul** | -0.026551 | 0.253823 | -0.629888 | 0.253823 | | **John** | 0.551427 | 0.745113 | -0.536573 | -0.897928 | |

#Resetting index

encrypted\_marks.reset\_index(inplace = True)

encrypted\_marks

|  |
| --- |
| output |
| | **index** | **Chemistry** | **Physics** | **Mathematics** | **English** | | --- | --- | --- | --- | --- | | **0** | Subodh | -0.855520 | 0.850904 | -0.262375 | 0.149877 | | **1** | Ram | 0.893997 | -0.779466 | -0.821818 | 0.893997 | | **2** | Abdul | -0.026551 | 0.253823 | -0.629888 | 0.253823 | | **3** | John | 0.551427 | 0.745113 | -0.536573 | -0.897928 |   In [73]: |

#The teacher wants to award five bonus marks to all the students.

new\_marks = marks\_df + 5

new\_marks

|  |
| --- |
| Output |
| | **Chemistry** | **Physics** | **Mathematics** | **English** | | --- | --- | --- | --- | | **Subodh** | 72 | 50 | 55 | 24 | | **Ram** | 95 | 97 | 92 | 95 | | **Abdul** | 71 | 77 | 86 | 77 | | **John** | 37 | 45 | 17 | 73 | |

#The teacher wants to increase the marks of all the students as follows-

#Chemistry: + 5

#Physics: + 10

#Mathematics: +10

#English: + 2

new\_marks = marks\_df + [5,10,10,2]

new\_marks

|  |
| --- |
| Output |
| | **Chemistry** | **Physics** | **Mathematics** | **English** | | --- | --- | --- | --- | | **Subodh** | 72 | 55 | 60 | 21 | | **Ram** | 95 | 102 | 97 | 92 | | **Abdul** | 71 | 82 | 91 | 74 | | **John** | 37 | 50 | 22 | 70 | |

#The teacher wants to get the total marks scored in each subject

marks\_df.apply(np.sum, axis = 0)

|  |
| --- |
| Output |
| Chemistry 255  Physics 249  Mathematics 230  English 249  dtype: int64 |

#The teacher wants to get the total marks scored by each student.

marks\_df.apply(np.sum, axis = 1)

|  |
| --- |
| Output |
| Subodh 181  Ram 359  Abdul 291  John 152  dtype: int64 |

#The students were unable to attend the next set of exams due to the pandemic. Hence, the teacher decides to award them average marks based on their previous performance.

marks\_df.apply(func = np.mean, axis = 0, result\_type = 'broadcast')

|  |
| --- |
| Output |
| | **Chemistry** | **Physics** | **Mathematics** | **English** | | --- | --- | --- | --- | | **Subodh** | 63 | 62 | 57 | 62 | | **Ram** | 63 | 62 | 57 | 62 | | **Abdul** | 63 | 62 | 57 | 62 | | **John** | 63 | 62 | 57 | 62 | |

#Consider the following tables of student marks belonging to different sections.

marks\_A = {'Chemistry': [67,90,66,32],

'Physics': [45,92,72,40],

}

marks\_A\_df = pd.DataFrame(marks\_A, index = ['Subodh', 'Ram', 'Abdul', 'John'])

marks\_B = {'Chemistry': [72,45,60,98],

'Physics': [78,34,72,95],

}

marks\_B\_df = pd.DataFrame(marks\_B, index = ['Nandini', 'Zoya', 'Shivam', 'James'])

#The teacher wants to combine the marks of these students.

pd.concat([marks\_A\_df,marks\_B\_df], sort = False)

|  |
| --- |
| Output |
| | **Chemistry** | **Physics** | | --- | --- | | **Subodh** | 67 | 45 | | **Ram** | 90 | 92 | | **Abdul** | 66 | 72 | | **John** | 32 | 40 | | **Nandini** | 72 | 78 | | **Zoya** | 45 | 34 | | **Shivam** | 60 | 72 | | **James** | 98 | 95 | |

df1 = pd.DataFrame({'employee': ['Jyoti', 'Sapna', 'Raj', 'Ramaswamy'],

'group': ['Accounting', 'Engineering', 'Engineering', 'HR']})

df2 = pd.DataFrame({'employee': ['Jyoti', 'Sapna', 'Raj', 'Ramaswamy'],

'hire\_date': [2004, 2008, 2012, 2014]})

display(df1,df2)

|  |
| --- |
| Output |
| | **employee** | **group** | | --- | --- | | **0** | Jyoti | Accounting | | **1** | Sapna | Engineering | | **2** | Raj | Engineering | | **3** | Ramaswamy | HR | |  | **employee** | **hire\_date** | | **0** | Jyoti | 2004 | | **1** | Sapna | 2008 | | **2** | Raj | 2012 | | **3** | Ramaswamy | 2014 | |

pd.concat([df1,df2], sort = False)

|  |
| --- |
| Output |
| | **employee** | **group** | **hire\_date** | | --- | --- | --- | | **0** | Jyoti | Accounting | NaN | | **1** | Sapna | Engineering | NaN | | **2** | Raj | Engineering | NaN | | **3** | Ramaswamy | HR | NaN | | **0** | Jyoti | NaN | 2004.0 | | **1** | Sapna | NaN | 2008.0 | | **2** | Raj | NaN | 2012.0 | | **3** | Ramaswamy | NaN | 2014.0 | |

# Using Merge in case of column mismatch

by default inner merge, use how for others

df3 = pd.merge(df1,df2)

df3

|  |
| --- |
| output |
| | **employee** | **group** | **hire\_date** | | --- | --- | --- | | **0** | Jyoti | Accounting | 2004 | | **1** | Sapna | Engineering | 2008 | | **2** | Raj | Engineering | 2012 | | **3** | Ramaswamy | HR | 2014 | |