**3. Pandas DataSeries:**

**i)Write a Pandas program to create and display a one-dimensional array-like object containing an array of data using Pandas module**.

**import pandas as pd**

**series = pd.Series([2, 4, 6, 8, 10])**

**print(series)**

**output:**

**0 2**

**1 4**

**2 6**

**3 8**

**4 10**

**dtype: int64**

**ii)Write a Pandas program to convert a Panda module Series to Python list and it's type.**

**import pandas as pd**

**ds = pd.Series([2, 4, 6, 8, 10])**

**print("Pandas Series and type")**

**print(ds)**

**print(type(ds))**

**print("Convert Pandas Series to Python list")**

**print(ds.tolist())**

**print(type(ds.tolist()))**

**output:**

**Pandas Series and type**

**0 2**

**1 4**

**2 6**

**3 8**

**4 10**

**dtype: int64**

**<class 'pandas.core.series.Series'>**

**Convert Pandas Series to Python list**

**[2, 4, 6, 8, 10]**

**<class 'list'>**

**iii)Write a Pandas program to add, subtract, multiple and divide two Pandas Series.**

**import pandas as pd**

**ds1 = pd.Series([2, 4, 6, 8, 10])**

**ds2 = pd.Series([1, 3, 5, 7, 9])**

**ds = ds1 + ds2**

**print("Add two Series:")**

**print(ds)**

**print("Subtract two Series:")**

**ds = ds1 - ds2**

**print(ds)**

**print("Multiply two Series:")**

**ds = ds1 \* ds2**

**print(ds)**

**print("Divide Series1 by Series2:")**

**ds = ds1 / ds2**

**print(ds)**

**output:**

**Add two Series:**

**0 3**

**1 7**

**2 11**

**3 15**

**4 19**

**dtype: int64**

**Subtract two Series:**

**0 1**

**1 1**

**2 1**

**3 1**

**4 1**

**dtype: int64**

**Multiply two Series:**

**0 2**

**1 12**

**2 30**

**3 56**

**4 90**

**dtype: int64**

**Divide Series1 by Series2:**

**0 2.000000**

**1 1.333333**

**2 1.200000**

**3 1.142857**

**4 1.111111**

**dtype: float64**

**iv)Write a Pandas program to convert a NumPy array to a Pandas series**

**import numpy as np**

**import pandas as pd**

**np\_array = np.array([10, 20, 30, 40, 50])**

**print("NumPy array:")**

**print(np\_array)**

**new\_series = pd.Series(np\_array)**

**print("Converted Pandas series:")**

**print(new\_series)**

**output**

**NumPy array:**

**[10 20 30 40 50]**

**Converted Pandas series:**

**0 10**

**1 20**

**2 30**

**3 40**

**4 50**

**dtype: int64**

**4. Pandas DataFrames:**

**i)Write a Pandas program to create and display a DataFrame from a specified dictionary data which has the index labels.**

**import pandas as pd**

**import numpy as np**

**exam\_data  = {'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael', 'Matthew', 'Laura', 'Kevin', 'Jonas'],**

**'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19],**

**'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],**

**'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']}**

**labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']**

**df = pd.DataFrame(exam\_data , index=labels)**

**print(df)**

**output:**

**name score attempts qualify**

**a Anastasia 12.5 1 yes**

**b Dima 9.0 3 no**

**c Katherine 16.5 2 yes**

**d James NaN 3 no**

**e Emily 9.0 2 no**

**f Michael 20.0 3 yes**

**g Matthew 14.5 1 yes**

**h Laura NaN 1 no**

**i Kevin 8.0 2 no**

**j Jonas 19.0 1 yes**

**ii)Write a Pandas program to change the name 'James' to 'Suresh' in name column of the DataFrame**

**import pandas as pd**

**import numpy as np**

**exam\_data  = {'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael', 'Matthew', 'Laura', 'Kevin', 'Jonas'],**

**'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19],**

**'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],**

**'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']}**

**labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']**

**df = pd.DataFrame(exam\_data , index=labels)**

**print("Original rows:")**

**print(df)**

**print("\nChange the name 'James' to ‘Suresh’:")**

**df['name'] = df['name'].replace('James', 'Suresh')**

**print(df)**

**output:**

**Original rows:**

**name score attempts qualify**

**a Anastasia 12.5 1 yes**

**b Dima 9.0 3 no**

**c Katherine 16.5 2 yes**

**d James NaN 3 no**

**e Emily 9.0 2 no**

**f Michael 20.0 3 yes**

**g Matthew 14.5 1 yes**

**h Laura NaN 1 no**

**i Kevin 8.0 2 no**

**j Jonas 19.0 1 yes**

**Change the name 'James' to ‘Suresh’:**

**name score attempts qualify**

**a Anastasia 12.5 1 yes**

**b Dima 9.0 3 no**

**c Katherine 16.5 2 yes**

**d Suresh NaN 3 no**

**e Emily 9.0 2 no**

**f Michael 20.0 3 yes**

**g Matthew 14.5 1 yes**

**h Laura NaN 1 no**

**i Kevin 8.0 2 no**

**j Jonas 19.0 1 yes**

**iii)Write a Pandas program to insert a new column in existing DataFrame**

**import pandas as pd**

**import numpy as np**

**exam\_data  = {'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael', 'Matthew', 'Laura', 'Kevin', 'Jonas'],**

**'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19],**

**'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],**

**'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']}**

**labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']**

**df = pd.DataFrame(exam\_data , index=labels)**

**print("Original rows:")**

**print(df)**

**color = ['Red','Blue','Orange','Red','White','White','Blue','Green','Green','Red']**

**df['color'] = color**

**print("\nNew DataFrame after inserting the 'color' column")**

**print(df)**

**output:**

**Original rows:**

**name score attempts qualify**

**a Anastasia 12.5 1 yes**

**b Dima 9.0 3 no**

**c Katherine 16.5 2 yes**

**d James NaN 3 no**

**e Emily 9.0 2 no**

**f Michael 20.0 3 yes**

**g Matthew 14.5 1 yes**

**h Laura NaN 1 no**

**i Kevin 8.0 2 no**

**j Jonas 19.0 1 yes**

**New DataFrame after inserting the 'color' column**

**name score attempts qualify color**

**a Anastasia 12.5 1 yes Red**

**b Dima 9.0 3 no Blue**

**c Katherine 16.5 2 yes Orange**

**d James NaN 3 no Red**

**e Emily 9.0 2 no White**

**f Michael 20.0 3 yes White**

**g Matthew 14.5 1 yes Blue**

**h Laura NaN 1 no Green**

**i Kevin 8.0 2 no Green**

**j Jonas 19.0 1 yes Red**

**iv)Write a Pandas program to get list from DataFrame column headers.**

**import pandas as pd**

**import numpy as np**

**exam\_data  = {'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael', 'Matthew', 'Laura', 'Kevin', 'Jonas'],**

**'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19],**

**'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],**

**'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']}**

**labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']**

**df = pd.DataFrame(exam\_data , index=labels)**

**print(list(df.columns.values))**

**output:**

**['name', 'score', 'attempts', 'qualify']**

**5. Pandas Index: i)Write a Pandas program to display the default index and set a column as an Index in a given dataframe.**

**import pandas as pd**

**df=pd.DataFrame({**

**'school\_code':['s001','s002','s003','s001','s002','s004'],**

**'class':['V','V','VI','VI','V','VI'],**

**'name':['Alberto Franco','GinoMcneill','Ryan Parkes','Eesha Hinton','Gino Mcneill','David Parkes'],**

**'date\_Of\_Birth':['15/05/2002','17/05/2002','16/02/1999','25/09/1998','11/05/2002','15/09/1997'],**

**'weight':[35,32,33,30,31,32],**

**'address':['street1','street2','street3','street1','street2','street4'],**

**'t\_id':['t1','t2','t3','t4','t5','t6']})**

**print("Default Index:")**

**print(df.head(10))**

**print("\nt\_id as new Index:")**

**df1 =df.set\_index('t\_id')**

**print(df1)**

**print("\nReset the index:")**

**df2 = df1.reset\_index(inplace=False)**

**print(df2)**

**output:**

**Default Index:**

**school\_code class name date\_Of\_Birth weight address t\_id**

**0 s001 V Alberto Franco 15/05/2002 35 street1 t1**

**1 s002 V Gino Mcneill 17/05/2002 32 street2 t2**

**2 s003 VI Ryan Parkes 16/02/1999 33 street3 t3**

**3 s001 VI Eesha Hinton 25/09/1998 30 street1 t4**

**4 s002 V Gino Mcneill 11/05/2002 31 street2 t5**

**5 s004 VI David Parkes 15/09/1997 32 street4 t6**

**t\_id as new Index:**

**school\_code class name date\_Of\_Birth weight address**

**t\_id**

**t1 s001 V Alberto Franco 15/05/2002 35 street1**

**t2 s002 V Gino Mcneill 17/05/2002 32 street2**

**t3 s003 VI Ryan Parkes 16/02/1999 33 street3**

**t4 s001 VI Eesha Hinton 25/09/1998 30 street1**

**t5 s002 V Gino Mcneill 11/05/2002 31 street2**

**t6 s004 VI David Parkes 15/09/1997 32 street4**

**Reset the index:**

**t\_idschool\_code class name date\_Of\_Birth weight address**

**0 t1 s001 V Alberto Franco 15/05/2002 35 street1**

**1 t2 s002 V Gino Mcneill 17/05/2002 32 street2**

**2 t3 s003 VI Ryan Parkes 16/02/1999 33 street3**

**3 t4 s001 VI Eesha Hinton 25/09/1998 30 street1**

**4 t5 s002 V Gino Mcneill 11/05/2002 31 street2**

**5 t6 s004 VI David Parkes 15/09/1997 32 street4**

**ii)Write a Pandas program to create an index labels by using 64-bit integers, using floating-point numbers in a given dataframe**

**import pandas as pd**

**print("Create an Int64Index:")**

**df\_i64 = pd.DataFrame({**

**'school\_code': ['s001','s002','s003','s001','s002','s004'],**

**'class': ['V', 'V', 'VI', 'VI', 'V', 'VI'],**

**'name': ['Alberto Franco','Gino Mcneill','Ryan Parkes', 'Eesha Hinton', 'Gino Mcneill', 'David Parkes'],**

**'date\_Of\_Birth': ['15/05/2002','17/05/2002','16/02/1999','25/09/1998','11/05/2002','15/09/1997'],**

**'weight': [35, 32, 33, 30, 31, 32],**

**'address': ['street1', 'street2', 'street3', 'street1', 'street2', 'street4']},**

**index=[1, 2, 3, 4, 5, 6])**

**print(df\_i64)**

**print("\nView the Index:")**

**print(df\_i64.index)**

**print("\nFloating-point labels using Float64Index:")**

**df\_f64 = pd.DataFrame({**

**'school\_code': ['s001','s002','s003','s001','s002','s004'],**

**'class': ['V', 'V', 'VI', 'VI', 'V', 'VI'],**

**'name': ['Alberto Franco','Gino Mcneill','Ryan Parkes', 'Eesha Hinton', 'Gino Mcneill', 'David Parkes'],**

**'date\_Of\_Birth ': ['15/05/2002','17/05/2002','16/02/1999','25/09/1998','11/05/2002','15/09/1997'],**

**'weight': [35, 32, 33, 30, 31, 32],**

**'address': ['street1', 'street2', 'street3', 'street1', 'street2', 'street4']},**

**index=[.1, .2, .3, .4, .5, .6])**

**print(df\_f64)**

**print("\nView the Index:")**

**print(df\_f64.index)**

**output:**

**Create an Int64Index:**

**school\_code class name date\_Of\_Birth weight address**

**1 s001 V Alberto Franco 15/05/2002 35 street1**

**2 s002 V Gino Mcneill 17/05/2002 32 street2**

**3 s003 VI Ryan Parkes 16/02/1999 33 street3**

**4 s001 VI Eesha Hinton 25/09/1998 30 street1**

**5 s002 V Gino Mcneill 11/05/2002 31 street2**

**6 s004 VI David Parkes 15/09/1997 32 street4**

**View the Index:**

**Int64Index([1, 2, 3, 4, 5, 6], dtype='int64')**

**Floating-point labels using Float64Index:**

**school\_code class name date\_Of\_Birth weight address**

**0.1 s001 V Alberto Franco 15/05/2002 35 street1**

**0.2 s002 V Gino Mcneill 17/05/2002 32 street2**

**0.3 s003 VI Ryan Parkes 16/02/1999 33 street3**

**0.4 s001 VI Eesha Hinton 25/09/1998 30 street1**

**0.5 s002 V Gino Mcneill 11/05/2002 31 street2**

**0.6 s004 VI David Parkes 15/09/1997 32 street4**

**View the Index:**

**Float64Index([0.1, 0.2, 0.3, 0.4, 0.5, 0.6], dtype='float64'**)

**6.Pandas Joining and merging DataFrame:**

**i)Write a Pandas program to join the two given dataframesalong rows and assign all data**.

**Program:**

import pandas as pd

student\_data1 = pd.DataFrame({

'student\_id': ['S1', 'S2', 'S3', 'S4', 'S5'],

'name': ['Danniella Fenton', 'Ryder Storey', 'Bryce Jensen', 'Ed Bernal', 'Kwame Morin'],

'marks': [200, 210, 190, 222, 199]})

student\_data2 = pd.DataFrame({

'student\_id': ['S4', 'S5', 'S6', 'S7', 'S8'],

'name': ['Scarlette Fisher', 'Carla Williamson', 'Dante Morse', 'Kaiser William', 'Madeeha Preston'],

'marks': [201, 200, 198, 219, 201]})

print("Original DataFrames:")

print(student\_data1)

print("-------------------------------------")

print(student\_data2)

print("\nJoin the said two dataframes along rows:")

result\_data = pd.concat([student\_data1, student\_data2])

print(result\_data)

**Output:**

Original DataFrames:

student\_id namemarks

0 S1Danniella Fenton 200

1 S2 Ryder Storey 210

2 S3 Bryce Jensen 190

3 S4 Ed Bernal 222

4 S5 Kwame Morin 199

-------------------------------------

student\_id name marks

0 S4 Scarlette Fisher 201

1 S5 Carla Williamson 200

2 S6 Dante Morse 198

3 S7 Kaiser William 219

4 S8 Madeeha Preston 201

Join the said two dataframes along rows:

student\_id name marks

0 S1 Danniella Fenton 200

1 S2 Ryder Storey 210

2 S3 Bryce Jensen 190

3 S4 Ed Bernal 222

4 S5 Kwame Morin 199

0 S4Scarlette Fisher 201

1 S5 Carla Williamson 200

2 S6 Dante Morse 198

3 S7 Kaiser William 219

4 S8 Madeeha Preston 201

**ii)Write a Pandas program to append a list of dictioneries or series to a existing DataFrame and display the combined data**

**Pogram:**

import pandas as pd

student\_data1 =pd.DataFrame({

'student\_id': ['S1', 'S2', 'S3', 'S4', 'S5'],

'name': ['Danniella Fenton', 'Ryder Storey', 'Bryce Jensen', 'Ed Bernal', 'Kwame Morin'],

'marks': [200, 210, 190, 222, 199]})

s6 = pd.Series(['S6', 'Scarlette Fisher', 205], index=['student\_id', 'name', 'marks'])

dicts = [{'student\_id': 'S6', 'name': 'Scarlette Fisher', 'marks': 203},

{'student\_id': 'S7', 'name': 'Bryce Jensen', 'marks': 207}]

print("Original DataFrames:")

print(student\_data1)

print("\nDictionary:")

print(s6)

combined\_data= student\_data1.append(dicts, ignore\_index=True, sort=False)

print("\nCombined Data:")

print(combined\_data)

**Output:**

Original DataFrames:

student\_id name marks

0 S1 Danniella Fenton 200

1 S2 Ryder Storey 210

2 S3 Bryce Jensen 190

3 S4 Ed Bernal 222

4 S5 Kwame Morin 199

Dictionary:

student\_id S6

nameScarlette Fisher

marks 205

dtype: object

Combined Data:

student\_id name marks

0 S1Danniella Fenton 200

1 S2 Ryder Storey 210

2 S3 Bryce Jensen 190

3 S4 Ed Bernal 222

4 S5 Kwame Morin 199

5 S6Scarlette Fisher 203

6 S7 Bryce Jensen 207

**iii)Write a Pandas program to join the two dataframes with matching records from both sides where available.**

**Program:**

import pandas as pd

student\_data1 = pd.DataFrame({

'student\_id': ['S1', 'S2', 'S3', 'S4', 'S5'],

'name': ['Danniella Fenton', 'Ryder Storey', 'Bryce Jensen', 'Ed Bernal', 'Kwame Morin'],

'marks': [200, 210, 190, 222, 199]})

student\_data2 = pd.DataFrame({

'student\_id': ['S4', 'S5', 'S6', 'S7', 'S8'],

'name': ['Scarlette Fisher', 'Carla Williamson', 'Dante Morse', 'Kaiser William', 'Madeeha Preston'],

'marks': [201, 200, 198, 219, 201]})

print("Original DataFrames:")

print(student\_data1)

print(student\_data2)

merged\_data = pd.merge(student\_data1, student\_data2, on='student\_id', how='outer')

print("Merged data (outer join):")

print(merged\_data)

**Output:**

Original DataFrames:

student\_id name marks

0 S1 Danniella Fenton 200

1 S2 Ryder Storey 210

2 S3 Bryce Jensen 190

3 S4 Ed Bernal 222

4 S5 Kwame Morin 199

student\_id name marks

0 S4 Scarlette Fisher 201

1 S5 Carla Williamson 200

2 S6 Dante Morse 198

3 S7 Kaiser William 219

4 S8 Madeeha Preston 201

Merged data (outer join):

student\_idname\_xmarks\_xname\_ymarks\_y

0 S1 Danniella Fenton 200.0 NaNNaN

1 S2 Ryder Storey 210.0 NaNNaN

2 S3 Bryce Jensen 190.0 NaNNaN

3 S4 Ed Bernal 222.0 Scarlette Fisher 201.0

4 S5 Kwame Morin 199.0 Carla Williamson 200.0

5 S6 NaNNaNDante Morse 198.0

6 S7 NaNNaNKaiser William 219.0

7 S8 NaNNaNMadeeha Preston 201.0

**7. Write a NumPy program to find the number of elements of an array, length of one array element in bytes and total bytes consumed by the elements.**

**Program:**

importnumpy as np

x = np.array([1,2,3], dtype=np.float64)

print("Size of the array: ", x.size)

print("Length of one array element in bytes: ", x.itemsize)

print("Total bytes consumed by the elements of the array: ", x.nbytes)

**Output:**

Size of the array: 3

Length of one array element in bytes: 8

Total bytes consumed by the elements of the array: 24

**8.Write a Pandas program to create  
a) Datetime object for Jan 15 2012.  
b) Specific date and time of 9:20 pm.  
c) Local date and time.  
d) A date without time.  
e) Current date.  
f) Time from a datetime.  
g) Current local time.**

**Program:**

importdatetime

fromdatetime import datetime

print("Datetime object for Jan 11 2012:")

print(datetime(2012, 1, 11))

print("\nSpecific date and time of 9:20 pm")

print(datetime(2011, 1, 11, 21, 20))

print("\nLocal date and time:")

print(datetime.now())

print("\nA date without time: ")

print(datetime.date(datetime(2012, 5, 22)))

print("\nCurrent date:")

print(datetime.now().date())

print("\nTime from a datetime:")

print(datetime.time(datetime(2012, 12, 15, 18, 12)))

print("\nCurrent local time:")

print(datetime.now().time())

**Output:**

Datetime object for Jan 11 2012:

2012-01-11 00:00:00

Specific date and time of 9:20 pm

2011-01-11 21:20:00

Local date and time:

2020-08-17 09:56:17.459790

A date without time:

2012-05-22

Current date:

2020-08-17

Time from a datetime:

18:12:00

Current local time:

09:56:17.461250

**EXPERIMENT-9:**

**i)Write a Pandas program to create a date from a given year, month, day and another date from a given string formats.**

**Program:**

fromdatetime import datetime

date1 = datetime(year=2020, month=12, day=25)

print("Date from a given year, month, day:")

print(date1)

fromdateutil import parser

date2 = parser.parse("1st of January, 2021")

print("\nDate from a given string formats:")

print(date2)

**Output:**

Date from a given year, month, day:

2020-12-25 00:00:00

Date from a given string formats:

2021-01-01 00:00:00

**ii)Write a Pandas program to create a time-series with two index labels and random values. Also print the type of the index.**

**Program:**

import pandas as pd

importnumpy as np

importdatetime

fromdatetime import datetime, date

dates = [datetime(2011, 9, 1), datetime(2011, 9, 2)]

print("Time-series with two index labels:")

time\_series = pd.Series(np.random.randn(2), dates)

print(time\_series)

print("\nType of the index:")

print(type(time\_series.index))

**Output:**

Time-series with two index labels:

2011-09-01 -0.257567

2011-09-02 0.947341

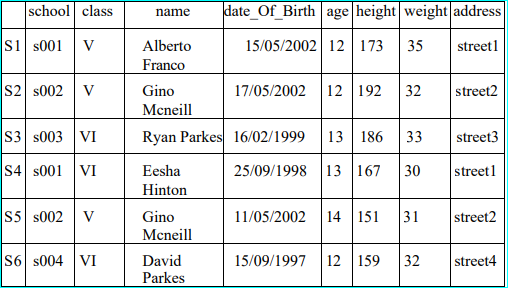
dtype: float64

Type of the index:

<class 'pandas.core.indexes.datetimes.DatetimeIndex'>

**10. Pandas Grouping Aggregate:**

**Consider dataset:**



**i)Write a Pandas program to split the following dataframe into groups based on school code. Also check the type of GroupBy object.**

**Program:**

import pandas as pd

pd.set\_option('display.max\_rows', None)

#pd.set\_option('display.max\_columns', None)

student\_data = pd.DataFrame({

'school\_code': ['s001','s002','s003','s001','s002','s004'],

'class': ['V', 'V', 'VI', 'VI', 'V', 'VI'],

'name': ['Alberto Franco','GinoMcneill','Ryan Parkes', 'Eesha Hinton', 'Gino Mcneill', 'David Parkes'],

'date\_Of\_Birth ': ['15/05/2002','17/05/2002','16/02/1999','25/09/1998','11/05/2002','15/09/1997'],

'age': [12, 12, 13, 13, 14, 12],

'height': [173, 192, 186, 167, 151, 159],

'weight': [35, 32, 33, 30, 31, 32],

'address': ['street1', 'street2', 'street3', 'street1', 'street2', 'street4']},

index=['S1', 'S2', 'S3', 'S4', 'S5', 'S6'])

print("Original DataFrame:")

print(student\_data)

print('\nSplit the said data on school\_code wise:')

result = student\_data.groupby(['school\_code'])

forname,group in result:

print("\nGroup:")

print(name)

print(group)

print("\nType of the object:")

print(type(result))

**Output:**

address

S1 s001 V Alberto Franco ... 173 35 street1

S4 s001 VI Eesha Hinton ... 167 30 street1

[2 rows x 8 columns]

Group:

s002

school\_code class name ... height weight address

S2 s002 V GinoMcneill ... 192 32 street2

S5 s002 V GinoMcneill ... 151 31 street2

[2 rows x 8 columns]

Group:

s003

school\_code class name ... height weight address

S3 s003 VI RyanParkes ... 186 33 street3

[1 rows x 8 columns]

Group:

s004

school\_code class name ... height weight address

S6 s004 VI DavidParkes ... 159 32 street4

[1 rows x 8 columns]

Type of the object:

<class 'pandas.core.groupby.groupby.DataFrameGroupBy'>

**ii)Write a Pandas program to split the following dataframe by school code and get mean, min, and max value of age for each school.**

**Program:**

import pandas as pd

pd.set\_option('display.max\_rows', None)

#pd.set\_option('display.max\_columns', None)

student\_data = pd.DataFrame({

'school\_code': ['s001','s002','s003','s001','s002','s004'],

'class': ['V', 'V', 'VI', 'VI', 'V', 'VI'],

'name': ['Alberto Franco','GinoMcneill','Ryan Parkes', 'Eesha Hinton', 'Gino Mcneill', 'David Parkes'],

'date\_Of\_Birth ': ['15/05/2002','17/05/2002','16/02/1999','25/09/1998','11/05/2002','15/09/1997'],

'age': [12, 12, 13, 13, 14, 12],

'height': [173, 192, 186, 167, 151, 159],

'weight': [35, 32, 33, 30, 31, 32],

'address': ['street1', 'street2', 'street3', 'street1', 'street2', 'street4']},

index=['S1', 'S2', 'S3', 'S4', 'S5', 'S6'])

print("Original DataFrame:")

print(student\_data)

print('\nMean, min, and max value of age for each value of the school:')

grouped\_single = student\_data.groupby('school\_code').agg({'age': ['mean', 'min', 'max']})

print(grouped\_single)

**Output:**

**Original DataFrame:**

school\_code class name ... height weight address

S1 s001 V Alberto Franco ... 173 35 street1

S2 s002 V Gino Mcneill ... 192 32 street2

S3 s003 VI Ryan Parkes ... 186 33 street3

S4 s001 VI Eesha Hinton ... 167 30 street1

S5 s002 V Gino Mcneill ... 151 31 street2

S6 s004 VI David Parkes ... 159 32 street4

[6 rows x 8 columns]

Mean, min, and max value of age for each value of the school:

age

meanminmax

school\_code

s001 12.5 12 13

s002 13.0 12 14

s003 13.0 13 13

s004 12.0 12 12

**11.**

**i) Create a dataframe of ten rows, four columns with random values. Write a Pandas program to highlight the negative numbers red and positive numbers black.**

**Pogram:**

import pandas as pd

import numpy as np

np.random.seed(24)

df = pd.DataFrame({'A': np.linspace(1, 10, 10)})

df = pd.concat([df, pd.DataFrame(np.random.randn(10, 4), columns=list('BCDE'))],

axis=1)

print("Original array:")

print(df)

defcolor\_negative\_red(val):

color = 'red' if val< 0 else 'black'

return 'color: %s' % color

print("\nNegative numbers red and positive numbers black:")

df.style.applymap(color\_negative\_red)

**Output:**

Original array:

A B C D E

0 1.0 1.329212 -0.770033 -0.316280 -0.990810

1 2.0 -1.070816 -1.438713 0.564417 0.295722

2 3.0 -1.626404 0.219565 0.678805 1.889273

3 4.0 0.961538 0.104011 -0.481165 0.850229

4 5.0 1.453425 1.057737 0.165562 0.515018

5 6.0 -1.336936 0.562861 1.392855 -0.063328

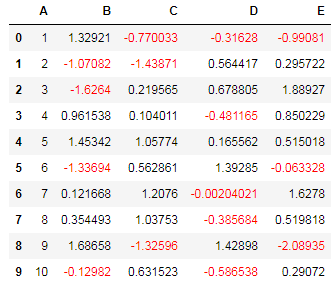
6 7.0 0.121668 1.207603 -0.002040 1.627796

7 8.0 0.354493 1.037528 -0.385684 0.519818

8 9.0 1.686583 -1.325963 1.428984 -2.089354

9 10.0 -0.129820 0.631523 -0.586538 0.290720

Negative numbers red and positive numbers black:



**ii) Create a dataframe of ten rows, four columns with random values. Write a Pandas program to highlight the maximum value in each column.**

**Program:**

import pandas as pd

import numpy as np

np.random.seed(24)

df = pd.DataFrame({'A': np.linspace(1, 10, 10)})

df = pd.concat([df, pd.DataFrame(np.random.randn(10, 4), columns=list('BCDE'))],

axis=1)

df.iloc[0, 2] = np.nan

df.iloc[3, 3] = np.nan

df.iloc[4, 1] = np.nan

df.iloc[9, 4] = np.nan

print("Original array:")

print(df)

defhighlight\_max(s):

'''

highlight the maximum in a Series green.

'''

is\_max = s == s.max()

return ['background-color: green' if v else '' for v in is\_max]

print("\nHighlight the maximum value in last two columns:")

df.style.apply(highlight\_max,subset=pd.IndexSlice[:, ['D', 'E']])

**Output:**

Original array:

A B C D E

0 1.0 1.329212 NaN -0.316280 -0.990810

1 2.0 -1.070816 -1.438713 0.564417 0.295722

2 3.0 -1.626404 0.219565 0.678805 1.889273

3 4.0 0.961538 0.104011 NaN0.850229

4 5.0 NaN 1.057737 0.165562 0.515018

5 6.0 -1.336936 0.562861 1.392855 -0.063328

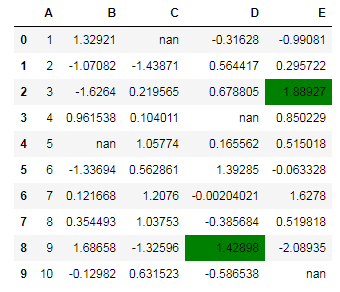
6 7.0 0.121668 1.207603 -0.002040 1.627796

7 8.0 0.354493 1.037528 -0.385684 0.519818

8 9.0 1.686583 -1.325963 1.428984 -2.089354

9 10.0 -0.129820 0.631523 -0.586538 NaN

Highlight the maximum value in last two columns:



**12. Create a dataframe of ten rows, four columns with random values. Write a Pandas program to highlightdataframe's specific columns.**

**Program:**

import pandas as pd

import numpy as np

np.random.seed(24)

df = pd.DataFrame({'A': np.linspace(1, 10, 10)})

df = pd.concat([df, pd.DataFrame(np.random.randn(10, 4), columns=list('BCDE'))],

axis=1)

df.iloc[0, 2] = np.nan

df.iloc[3, 3] = np.nan

df.iloc[4, 1] = np.nan

df.iloc[9, 4] = np.nan

print("Original array:")

print(df)

defhighlight\_cols(s):

color = 'grey'

return 'background-color: %s' % color

print("\nHighlight specific columns:")

df.style.applymap(highlight\_cols, subset=pd.IndexSlice[:, ['B', 'C']])

**Output:**

A B C D E

0 1.0 1.329212 NaN-0.316280 -0.990810

1 2.0 -1.070816 -1.438713 0.564417 0.295722

2 3.0 -1.626404 0.219565 0.678805 1.889273

3 4.0 0.961538 0.104011 NaN 0.850229

4 5.0 NaN 1.057737 0.165562 0.515018

5 6.0 -1.336936 0.562861 1.392855 -0.063328

6 7.0 0.121668 1.207603 -0.002040 1.627796

7 8.0 0.354493 1.037528 -0.385684 0.519818

8 9.0 1.686583 -1.325963 1.428984 -2.089354

9 10.0 -0.129820 0.631523 -0.586538 NaN

Highlight specific columns:

