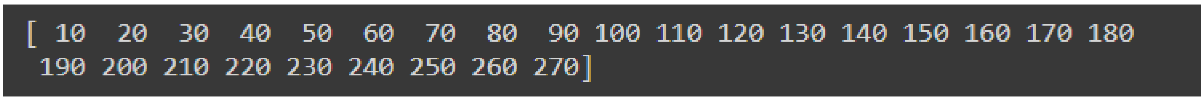
DSA0508-QUERY PROCESSING

LAB EXERCISES

1. **Write a Pandas program to select distinct department id from employees file.**

```
import pandas as pd
data = {
    'DEPARTMENT_ID': [10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120, 130, 140, 150, 160, 170, 180, 190, 200, 210, 220, 230, 240, 250, 260, 270],
    'DEPARTMENT_NAME': ['Administration', 'Marketing', 'Purchasing', 'Human Resources', 'Shipping', 'IT', 'Public Relations', 'Sales', 'Executive', 'Finance', 'Accounting', 'Treasury', 'Corporate Tax', 'Control And Credit', 'Shareholder Services', 'Benefits', 'Manufacturing', 'Construction', 'Contracting', 'Operations', 'IT Support', 'NOC', 'IT Helpdesk', 'Government Sales', 'Retail Sales', 'Recruiting', 'Payroll'],
    'MANAGER_ID': [200, 201, 114, 203, 121, 103, 204, 145, 100, 108, 205, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'LOCATION_ID': [1700, 1800, 1700, 2400, 1500, 1400, 2700, 2500, 1700, 1700, 1700, 1700, 1700, 1700, 1700, 1700, 1700, 1700, 1700, 1700, 1700, 1700, 1700, 1700, 1700, 1700, 1700]
}
employees_df = pd.DataFrame(data)
distinct_department_ids = employees_df['DEPARTMENT_ID'].unique()
print(distinct_department_ids)
```

OUTPUT:

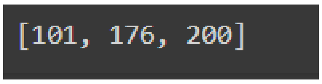


```
[ 10  20  30  40  50  60  70  80  90 100 110 120 130 140 150 160 170 180
 190 200 210 220 230 240 250 260 270]
```

2. **Write a Pandas program to display the ID for those employees who did two or more jobs in the past.**

```
 import pandas as pd
data = {
    'EMPLOYEE_ID': [102, 101, 101, 201, 114, 122, 200, 176, 176, 200],
    'START_DATE': ['2001-01-13', '1997-09-21', '2001-10-28', '2004-02-17', '2006-03-24', '2007-01-01', '1995-09-17', '2006-03-24', '2007-01-01', '2002-07-01'],

    'END_DATE': ['2006-07-24', '2001-10-27', '2005-03-15', '2007-12-19', '2007-12-31', '2007-12-31', '2001-06-17', '2006-12-31', '2007-12-31', '2006-12-31'],

    'JOB_ID': ['IT_PROG', 'AC_ACCOUNT', 'AC_MGR', 'MK_REP', 'ST_CLERK', 'ST_CLERK', 'AD_ASST', 'SA_REP', 'SA_MAN', 'AC_ACCOUNT'],

    'DEPARTMENT_ID': [60, 110, 110, 20, 50, 50, 90, 80, 80, 90]

}

employees_df = pd.DataFrame(data)

employee_jobs_count = employees_df.groupby('EMPLOYEE_ID')['JOB_ID'].nunique()

employees_with_multiple_jobs = employee_jobs_count[employee_jobs_count >= 2]

print(employees_with_multiple_jobs.index.tolist())
```

OUTPUT:



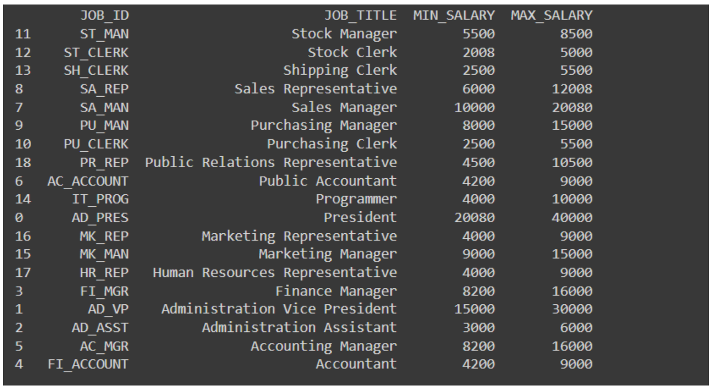
```
[101, 176, 200]
```

3. **Write a Pandas program to display the details of jobs in descending sequence on job title.**

```
import pandas as pd

data = {

    'JOB_ID': ['AD_PRES', 'AD_VP', 'AD_ASST', 'FI_MGR', 'FI_ACCOUNT', 'AC_MGR', 'AC_ACCOUNT', 'SA_MAN', 'SA_REP', 'PU_MAN', 'PU_CLERK', 'ST_MAN', 'ST_CLERK', 'SH_CLERK', 'IT_PROG', 'MK_MAN', 'MK_REP', 'HR_REP', 'PR_REP'],

    'JOB_TITLE': ['President', 'Administration Vice President', 'Administration Assistant', 'Finance Manager', 'Accountant', 'Accounting Manager', 'Public Accountant', 'Sales Manager', 'Sales Representative', 'Purchasing Manager', 'Purchasing Clerk', 'Stock Manager', 'Stock Clerk', 'Shipping Clerk', 'Programmer', 'Marketing Manager', 'Marketing Representative', 'Human Resources Representative', 'Public Relations Representative'],

    'MIN_SALARY': [20080, 15000, 3000, 8200, 4200, 8200, 4200, 10000, 6000, 8000, 2500, 5500, 2008, 2500, 4000, 9000, 4000, 4000, 4500],

    'MAX_SALARY': [40000, 30000, 6000, 16000, 9000, 16000, 9000, 20080, 12008, 15000, 5500, 8500, 5000, 5500, 10000, 15000, 9000, 9000, 10500]

}

jobs_df = pd.DataFrame(data)
```

```
sorted_jobs_df = jobs_df.sort_values(by='JOB_TITLE', ascending=False)
print(sorted_jobs_df)
```

OUTPUT:

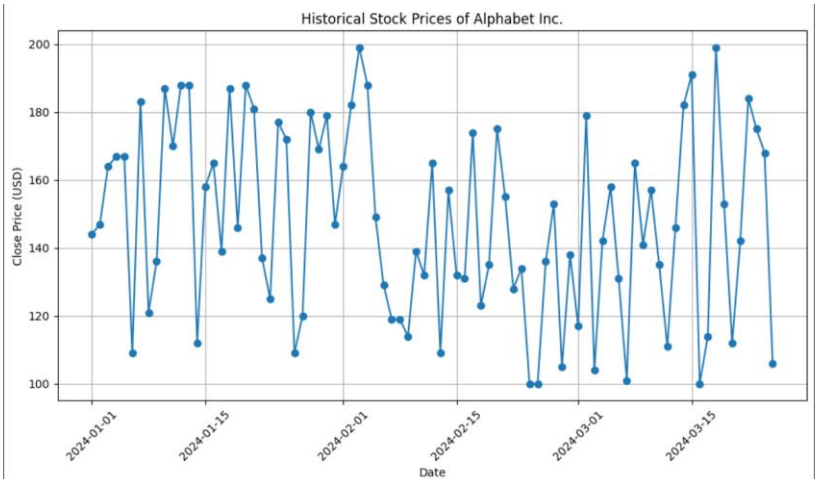


```
        JOB_ID                        JOB_TITLE  MIN_SALARY  MAX_SALARY
11      ST_MAN                    Stock Manager        5500        8500
12    ST_CLERK                      Stock Clerk        2008        5000
13    SH_CLERK                   Shipping Clerk        2500        5500
8       SA_REP             Sales Representative        6000       12008
7       SA_MAN                    Sales Manager       10000       20080
9       PU_MAN               Purchasing Manager        8000       15000
10    PU_CLERK                 Purchasing Clerk        2500        5500
18      PR_REP  Public Relations Representative        4500       10500
6   AC_ACCOUNT                Public Accountant        4200        9000
14     IT_PROG                       Programmer        4000       10000
0      AD_PRES                        President       20080       40000
16      MK_REP         Marketing Representative        4000        9000
15      MK_MAN                Marketing Manager        9000       15000
17      HR_REP   Human Resources Representative        4000        9000
3       FI_MGR                  Finance Manager        8200       16000
1        AD_VP     Administration Vice President       15000       30000
2      AD_ASST         Administration Assistant        3000        6000
5       AC_MGR               Accounting Manager        8200       16000
4   FI_ACCOUNT                       Accountant        4200        9000
```

4. Write a Pandas program to create a line plot of the historical stock prices of Alphabet Inc. between two specific dates.

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
start_date = '2024-01-01'
end_date = '2024-03-25'
dates = pd.date_range(start=start_date, end=end_date)
num_days = len(dates)
np.random.seed(0)
stock_prices = np.random.randint(100, 200, size=num_days).astype(float)
df = pd.DataFrame({'Date': dates, 'Close': stock_prices})
plt.figure(figsize=(10, 6))
plt.plot(df['Date'], df['Close'], marker='o', linestyle='-')
plt.title('Historical Stock Prices of Alphabet Inc.')
plt.xlabel('Date')
plt.ylabel('Close Price (USD)')
plt.grid(True)
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

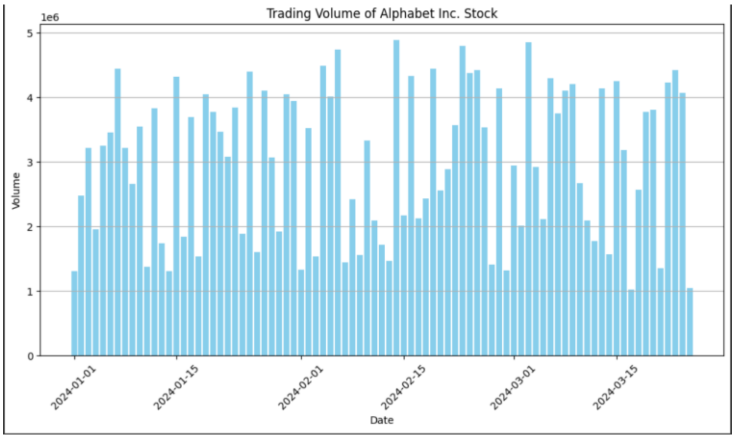
OUTPUT:



5. **Write a Pandas program to create a bar plot of the trading volume of Alphabet Inc. stock between two specific dates.**

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
start_date = '2024-01-01'
end_date = '2024-03-25'
dates = pd.date_range(start=start_date, end=end_date)
num_days = len(dates)
np.random.seed(0)
trading_volume = np.random.randint(1000000, 5000000, size=num_days)
df = pd.DataFrame({'Date': dates, 'Volume': trading_volume})
plt.figure(figsize=(10, 6))
plt.bar(df['Date'], df['Volume'], color='skyblue')
plt.title('Trading Volume of Alphabet Inc. Stock')
plt.xlabel('Date')
plt.ylabel('Volume')
plt.grid(axis='y')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

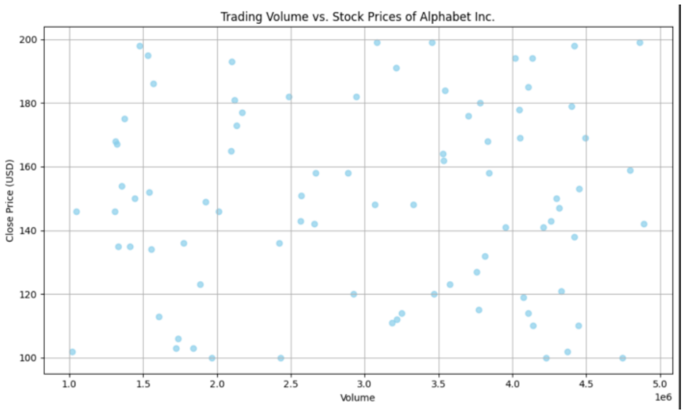
OUTPUT:



6. **Write a Pandas program to create a scatter plot of the trading volume/stock prices of Alphabet Inc. stock between two specific dates.**

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
start_date = '2024-01-01'
end_date = '2024-03-25'
dates = pd.date_range(start=start_date, end=end_date)
num_days = len(dates)
np.random.seed(0)
trading_volume = np.random.randint(1000000, 5000000, size=num_days)
stock_prices = np.random.randint(100, 200, size=num_days)
df = pd.DataFrame({'Date': dates, 'Volume': trading_volume, 'Close': stock_prices})
plt.figure(figsize=(10, 6))
plt.scatter(df['Volume'], df['Close'], color='skyblue', alpha=0.7)
plt.title('Trading Volume vs. Stock Prices of Alphabet Inc.')
plt.xlabel('Volume')
plt.ylabel('Close Price (USD)')
plt.grid(True)
plt.tight_layout()
plt.show()
```

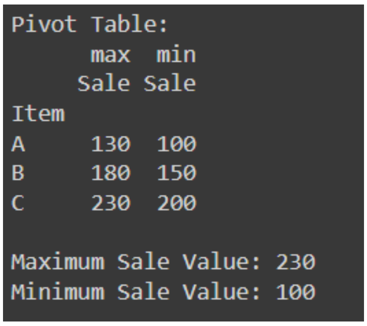
OUTPUT:



7. **Write a Pandas program to create a Pivot table and find the maximum and minimum sale value of the items.**

```
import pandas as pd
sales_data = {
    'Item': ['A', 'B', 'C', 'A', 'B', 'C', 'A', 'B', 'C'],
    'Date': ['2024-01-01', '2024-01-01', '2024-01-01', '2024-01-02', '2024-01-02', '2024-01-02', '2024-01-03', '2024-01-03', '2024-01-03'],
    'Sale': [100, 150, 200, 120, 170, 220, 130, 180, 230]
}
df = pd.DataFrame(sales_data)
pivot_table = pd.pivot_table(df, values='Sale', index='Item', aggfunc=['max', 'min'])
print("Pivot Table:")
print(pivot_table)
max_sale_value = pivot_table['max'].max().iloc[0]
min_sale_value = pivot_table['min'].min().iloc[0]
print("\nMaximum Sale Value:", max_sale_value)
print("Minimum Sale Value:",min_sale_value)
```

OUTPUT:

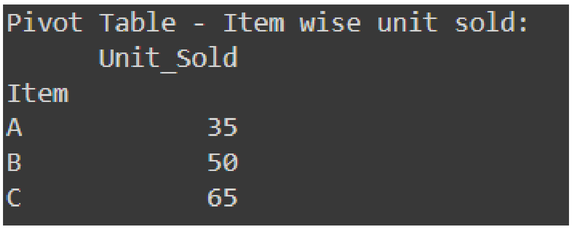


```
Pivot Table:
      max  min
     Sale Sale
Item
A     130  100
B     180  150
C     230  200

Maximum Sale Value: 230
Minimum Sale Value: 100
```

8. **Write a Pandas program to create a Pivot table and find the item wise unit sold.**

```
import pandas as pd
sales_data = {
    'Item': ['A', 'B', 'C', 'A', 'B', 'C', 'A', 'B', 'C'],
    'Date': ['2024-01-01', '2024-01-01', '2024-01-01', '2024-01-02', '2024-01-02', '2024-01-02', '2024-01-03', '2024-01-03', '2024-01-03'],
    'Unit_Sold': [10, 15, 20, 12, 17, 22, 13, 18, 23]
}
df = pd.DataFrame(sales_data)
pivot_table = pd.pivot_table(df, values='Unit_Sold', index='Item', aggfunc='sum')
print("Pivot Table - Item wise unit sold:")
print(pivot_table)
```

OUTPUT:



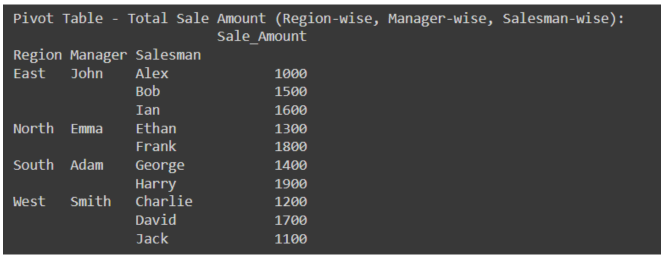
```
Pivot Table - Item wise unit sold:
      Unit_Sold
Item
A            35
B            50
C            65
```

9. Write a Pandas program to create a Pivot table and find the total sale

amount region wise, manager wise, sales man wise.

```
import pandas as pd

sales_data = {

    'Region': ['East', 'East', 'West', 'West', 'North', 'North', 'South', 'South', 'East', 'West'],

    'Manager': ['John', 'John', 'Smith', 'Smith', 'Emma', 'Emma', 'Adam', 'Adam', 'John', 'Smith'],

    'Salesman': ['Alex', 'Bob', 'Charlie', 'David', 'Ethan', 'Frank', 'George', 'Harry', 'Ian', 'Jack'],

    'Sale_Amount': [1000, 1500, 1200, 1700, 1300, 1800, 1400, 1900, 1600, 1100]

}

df = pd.DataFrame(sales_data)

pivot_table = pd.pivot_table(df, values='Sale_Amount', index=['Region', 'Manager', 'Salesman'], aggfunc='sum')

print("Pivot Table - Total Sale Amount (Region-wise, Manager-wise, Salesman-wise):")

print(pivot_table)
```

OUTPUT:

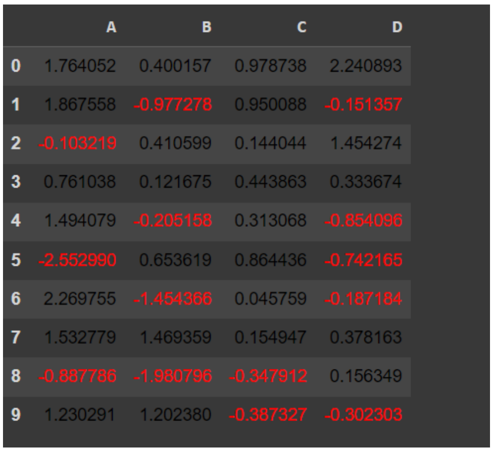


```
Pivot Table - Total Sale Amount (Region-wise, Manager-wise, Salesman-wise):
                         Sale_Amount
Region Manager Salesman
East   John    Alex             1000
               Bob              1500
               Ian              1600
North  Emma    Ethan            1300
               Frank            1800
South  Adam    George           1400
               Harry            1900
West   Smith   Charlie          1200
               David            1700
               Jack             1100
```

10. **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.**

```
import pandas as pd

import numpy as np

np.random.seed(0)

data = np.random.randn(10, 4)

df = pd.DataFrame(data, columns=['A', 'B', 'C', 'D'])

def color_negative_red(val):

    color = 'red' if val < 0 else 'black'
    return f'color: {color}'
styled_df = df.style.applymap(color_negative_red)
styled_df
```

OUTPUT:



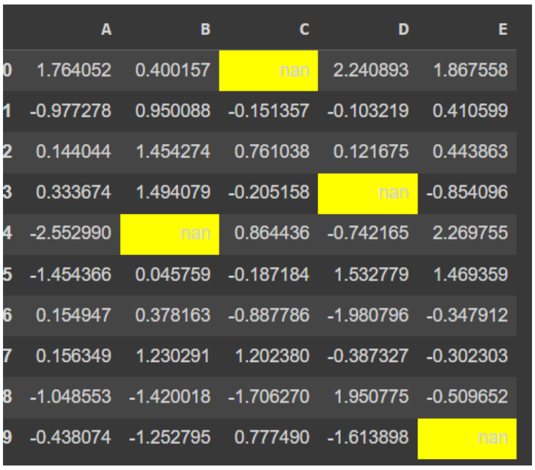
	A	B	C	D
0	1.764052	0.400157	0.978738	2.240893
1	1.867558	-0.977278	0.950088	-0.151357
2	-0.103219	0.410599	0.144044	1.454274
3	0.761038	0.121675	0.443863	0.333674
4	1.494079	-0.205158	0.313068	-0.854096
5	-2.552990	0.653619	0.864436	-0.742165
6	2.269755	-1.454366	0.045759	-0.187184
7	1.532779	1.469359	0.154947	0.378163
8	-0.887786	-1.980796	-0.347912	0.156349
9	1.230291	1.202380	-0.387327	-0.302303

11.**Create a dataframe of ten rows, four columns with random values. Convert some values to nan values. Write a Pandas program which will highlight the nan values.**

```
import pandas as pd
import numpy as np
np.random.seed(0)
data = np.random.randn(10, 5)
df = pd.DataFrame(data, columns=['A', 'B', 'C', 'D', 'E'])
df.loc[4:4, 'B'] = np.nan
df.loc[3:3, 'D'] = np.nan
df.loc[0:0, 'C'] = np.nan
df.loc[9:9, 'E'] = np.nan
def highlight_nan(val):
    if pd.isna(val):
        return 'background-color: yellow'
    else:
        return ''
```

```
styled_df = df.style.applymap(highlight_nan)
styled_df
```

OUTPUT:

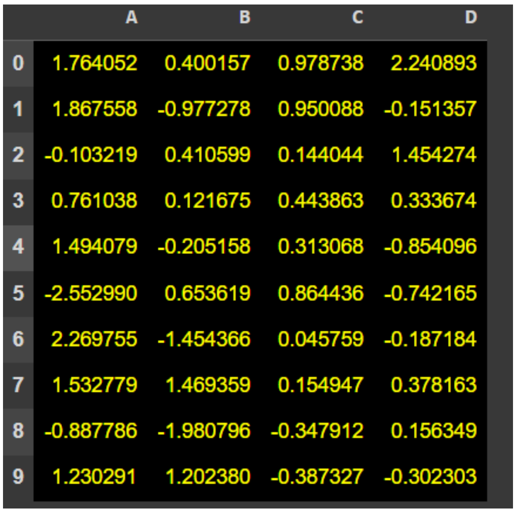


	A	B	C	D	E
0	1.764052	0.400157	nan	2.240893	1.867558
1	-0.977278	0.950088	-0.151357	-0.103219	0.410599
2	0.144044	1.454274	0.761038	0.121675	0.443863
3	0.333674	1.494079	-0.205158	nan	-0.854096
4	-2.552990	nan	0.864436	-0.742165	2.269755
5	-1.454366	0.045759	-0.187184	1.532779	1.469359
6	0.154947	0.378163	-0.887786	-1.980796	-0.347912
7	0.156349	1.230291	1.202380	-0.387327	-0.302303
8	-1.048553	-1.420018	-1.706270	1.950775	-0.509652
9	-0.438074	-1.252795	0.777490	-1.613898	nan

12.Create a dataframe of ten rows, four columns with random values. Write a Pandas program to set dataframe background Color black and font color yellow.

```
import pandas as pd
import numpy as np
np.random.seed(0)
data = np.random.randn(10, 4)
df = pd.DataFrame(data, columns=['A', 'B', 'C', 'D'])
def set_colors(val):
    return 'background-color: black; color: yellow'
styled_df = df.style.applymap(lambda x: set_colors(x))
styled_df
```

OUTPUT:

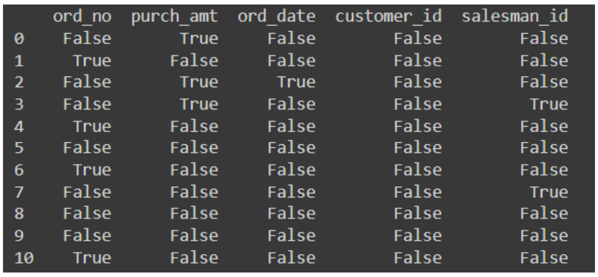


	A	B	C	D
0	1.764052	0.400157	0.978738	2.240893
1	1.867558	-0.977278	0.950088	-0.151357
2	-0.103219	0.410599	0.144044	1.454274
3	0.761038	0.121675	0.443863	0.333674
4	1.494079	-0.205158	0.313068	-0.854096
5	-2.552990	0.653619	0.864436	-0.742165
6	2.269755	-1.454366	0.045759	-0.187184
7	1.532779	1.469359	0.154947	0.378163
8	-0.887786	-1.980796	-0.347912	0.156349
9	1.230291	1.202380	-0.387327	-0.302303

13.Write a Pandas program to detect missing values of a given DataFrame. Display True or False.

```
import pandas as pd
import numpy as np
data = {
    'ord_no': [70001.0, np.nan, 70002.0, 70004.0, np.nan, 70005.0, np.nan, 70010.0, 70003.0, 70012.0, np.nan],
    'purch_amt': [None, 150.50, None, None, 65.26, 110.50, 270.65, 1983.43, 2480.40, 250.45, 75.29],
    'ord_date': ['2012-10-05', '2012-09-10', None, '2012-09-10', '2012-08-17', '2012-09-10', '2012-07-27', '2012-09-10', '2012-10-10', '2012-06-27', '2012-08-17'],
    'customer_id': [3002, 3001, 3001, 3003, 3002, 3001, 3001, 3004, 3003, 3002, 3001],
    'salesman_id': [5002.0, 5002.0, 5003.0, np.nan, 5002.0, 5003.0, 5001.0, np.nan, 5003.0, 5002.0, 5003.0]
}

df = pd.DataFrame(data)
missing_values = df.isna()
print(missing_values)
```

OUTPUT:

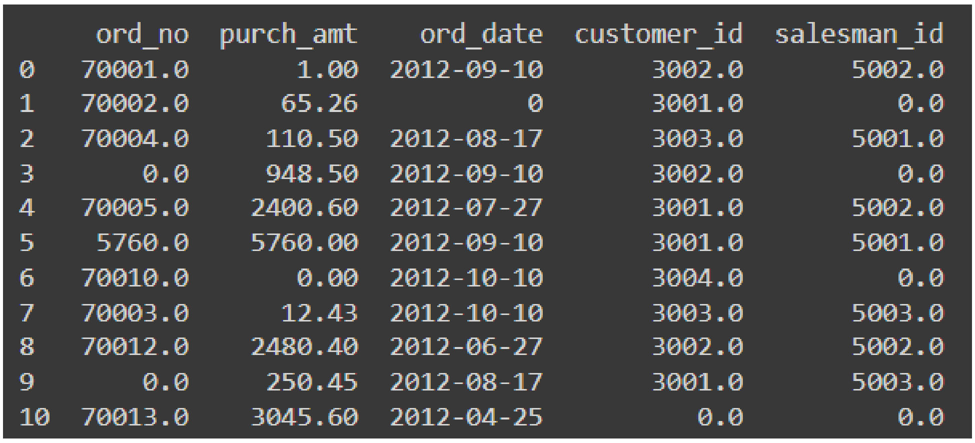


```
    ord_no  purch_amt  ord_date  customer_id  salesman_id
0    False       True     False        False        False
1     True      False     False        False        False
2    False       True      True        False        False
3    False       True     False        False         True
4     True      False     False        False        False
5    False      False     False        False        False
6     True      False     False        False        False
7    False      False     False        False         True
8    False      False     False        False        False
9    False      False     False        False        False
10    True      False     False        False        False
```

14. **Write a Pandas program to find and replace the missing values in a given DataFrame which do not have any valuable information.**

```
import pandas as pd

import numpy as np

data = {

    'ord_no': [70001, 70002, 70004, np.nan, 70005, 5760, 70010, 70003, 70012, np.nan, 
70013],

    'purch_amt': [1, 65.26, 110.5, 948.5, 2400.6, 5760, '?', 12.43, 2480.4, 250.45, 3045.6],

    'ord_date': ['2012-09-10', np.nan, '2012-08-17', '2012-09-10', '2012-07-27', '2012-09-10', 
'2012-10-10', '2012-10-10', '2012-06-27', '2012-08-17', '2012-04-25'],

    'customer_id': [3002, 3001, 3003, 3002, 3001, 3001, 3004, 3003, 3002, 3001, np.nan],

    'salesman_id': [5002, np.nan, 5001, np.nan, 5002, 5001, np.nan, 5003, 5002, 5003, np.nan]

}


df = pd.DataFrame(data)

df.replace('?', np.nan, inplace=True)

df.fillna(0, inplace=True)

print(df)
```

OUTPUT:



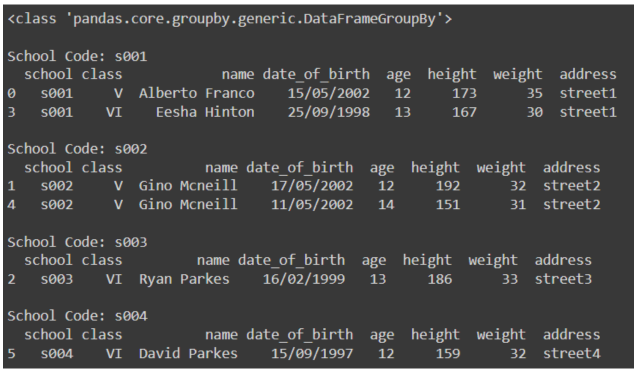
```
     ord_no  purch_amt    ord_date  customer_id  salesman_id
0   70001.0       1.00  2012-09-10       3002.0       5002.0
1   70002.0      65.26           0       3001.0          0.0
2   70004.0     110.50  2012-08-17       3003.0       5001.0
3       0.0     948.50  2012-09-10       3002.0          0.0
4   70005.0    2400.60  2012-07-27       3001.0       5002.0
5    5760.0    5760.00  2012-09-10       3001.0       5001.0
6   70010.0       0.00  2012-10-10       3004.0          0.0
7   70003.0      12.43  2012-10-10       3003.0       5003.0
8   70012.0    2480.40  2012-06-27       3002.0       5002.0
9       0.0     250.45  2012-08-17       3001.0       5003.0
10  70013.0    3045.60  2012-04-25          0.0          0.0
```

15. Write a Pandas program to keep the rows with at least 2 NaN values in a given DataFrame.

```
import pandas as pd

data = {

   'school': ['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'],

   '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']

}


df = pd.DataFrame(data)


# Group the DataFrame by school code

grouped = df.groupby('school')


# Check the type of GroupBy object

print(type(grouped))
```

```
# Iterate over the groups and display them
for name, group in grouped:
    print("\nSchool Code:", name)
    print( group)
```

OUTPUT:



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

School Code: s001
  school class           name date_of_birth  age  height  weight  address
0   s001     V  Alberto Franco    15/05/2002   12     173      35  street1
3   s001    VI    Eesha Hinton    25/09/1998   13     167      30  street1

School Code: s002
  school class          name date_of_birth  age  height  weight  address
1   s002     V  Gino Mcneill    17/05/2002   12     192      32  street2
4   s002     V  Gino Mcneill    11/05/2002   14     151      31  street2

School Code: s003
  school class         name date_of_birth  age  height  weight  address
2   s003    VI  Ryan Parkes    16/02/1999   13     186      33  street3

School Code: s004
  school class          name date_of_birth  age  height  weight  address
5   s004    VI  David Parkes    15/09/1997   12     159      32  street4
```

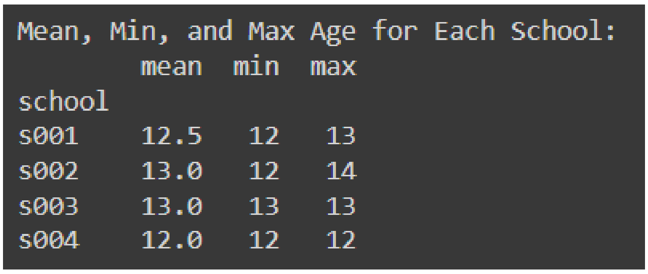
16.Write a Pandas program to split the following dataframe into groups based on school code. Also check the type of GroupBy object.

```
import pandas as pd
data = {
    'school': ['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'],
    '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']
}

df = pd.DataFrame(data)
result = df.groupby('school')['age'].agg(['mean', 'min', 'max'])
```

print("Mean, Min, and Max Age for Each School:")

print(result)

OUTPUT:

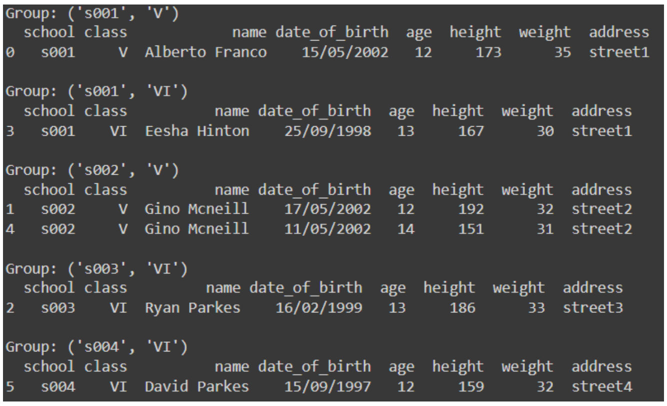


```
Mean, Min, and Max Age for Each School:
        mean  min  max
school
s001    12.5   12   13
s002    13.0   12   14
s003    13.0   13   13
s004    12.0   12   12
```

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

```
import pandas as pd
data = {
    'school': ['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'],
    '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']
}

df = pd.DataFrame(data)
grouped = df.groupby(['school', 'class'])
for name, group in grouped:
    print("\nGroup:", name)
    print(group)
```

OUTPUT:

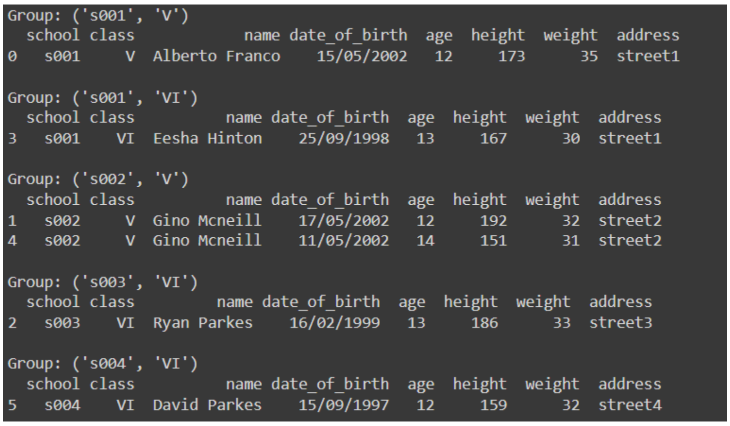


```
Group: ('s001', 'V')
  school class            name date_of_birth  age  height  weight  address
0   s001     V  Alberto Franco    15/05/2002   12     173      35  street1

Group: ('s001', 'VI')
  school class          name date_of_birth  age  height  weight  address
3   s001    VI  Eesha Hinton    25/09/1998   13     167      30  street1

Group: ('s002', 'V')
  school class          name date_of_birth  age  height  weight  address
1   s002     V  Gino Mcneill    17/05/2002   12     192      32  street2
4   s002     V  Gino Mcneill    11/05/2002   14     151      31  street2

Group: ('s003', 'VI')
  school class         name date_of_birth  age  height  weight  address
2   s003    VI  Ryan Parkes    16/02/1999   13     186      33  street3

Group: ('s004', 'VI')
  school class          name date_of_birth  age  height  weight  address
5   s004    VI  David Parkes    15/09/1997   12     159      32  street4
```

18.Write a Pandas program to split the following given dataframe into groups based on school code and class.

```
import pandas as pd

data = {

    'school': ['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'],

    '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']

}


df = pd.DataFrame(data)

grouped = df.groupby(['school', 'class'])

for name, group in grouped:

    print("\nGroup:", name)

    print(group)
```

OUTPUT:

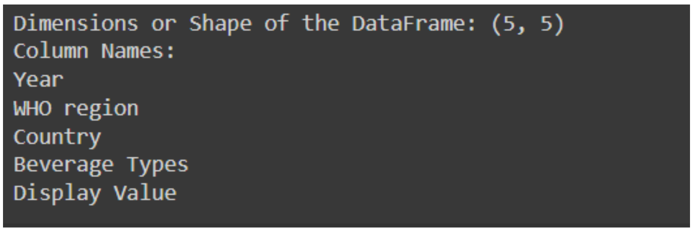


```
Group: ('s001', 'V')
  school class           name date_of_birth  age  height  weight  address
0   s001     V  Alberto Franco    15/05/2002   12     173      35  street1

Group: ('s001', 'VI')
  school class         name date_of_birth  age  height  weight  address
3   s001    VI  Eesha Hinton    25/09/1998   13     167      30  street1

Group: ('s002', 'V')
  school class         name date_of_birth  age  height  weight  address
1   s002     V  Gino Mcneill    17/05/2002   12     192      32  street2
4   s002     V  Gino Mcneill    11/05/2002   14     151      31  street2

Group: ('s003', 'VI')
  school class        name date_of_birth  age  height  weight  address
2   s003    VI  Ryan Parkes    16/02/1999   13     186      33  street3

Group: ('s004', 'VI')
  school class         name date_of_birth  age  height  weight  address
5   s004    VI  David Parkes    15/09/1997   12     159      32  street4
```

19.Write a Pandas program to display the dimensions or shape of the World alcohol consumption dataset. Also extract the column names from the dataset.

```
import pandas as pd
data = {
    'Year': [1986, 1986, 1985, 1986, 1987],
    'WHO region': ['Western Pacific', 'Americas', 'Africa', 'Americas', 'Americas'],
    'Country': ['Viet Nam', 'Uruguay', "Cte d'Ivoire", 'Colombia', 'Saint Kitts and Nevis'],
    'Beverage Types': ['Wine', 'Other', 'Wine', 'Beer', 'Beer'],
    'Display Value': [0.00, 0.50, 1.62, 4.27, 1.98]
}

df = pd.DataFrame(data)
print("Dimensions or Shape of the DataFrame:", df.shape)
column_names = df.columns.tolist()
print("Column Names:")
for name in column_names:
  print(name)
```

OUTPUT:



```
Dimensions or Shape of the DataFrame: (5, 5)
Column Names:
Year
WHO region
Country
Beverage Types
Display Value
```

20. Write a Pandas program to find the index of a given substring of a DataFrame column.

```
import pandas as pd
data = {
    'Column': ['apple', 'banana', 'orange', 'grape', 'watermelon']
}

df = pd.DataFrame(data)
substring = 'ran'
index = df[df['Column'].str.contains(substring)].index.tolist()
print("Index of the substring '{}' in the DataFrame column:".format(substring),index)
```

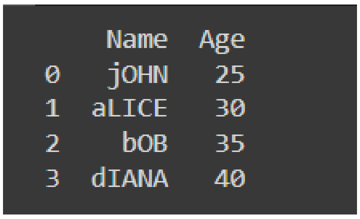
OUTPUT:

Index of the substring 'ran' in the DataFrame column: [2]

21.Write a Pandas program to swap the cases of a specified character column in a given DataFrame.

```
import pandas as pd
data = {'Name': ['John', 'Alice', 'Bob', 'Diana'],
        'Age': [25, 30, 35, 40]}
df = pd.DataFrame(data)
df['Name'] = df['Name'].str.swapcase()
print(df)
```

OUTPUT:

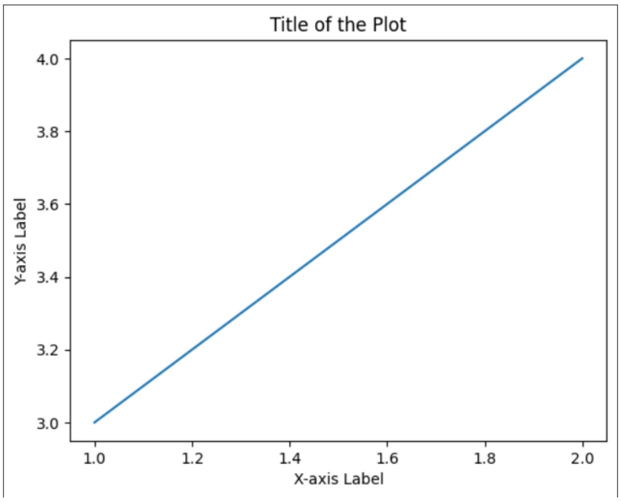


```
    Name  Age
0   jOHN   25
1  aLICE   30
2    bOB   35
3  dIANA   40
```

22. Write a Python program to draw a line with suitable label in the x axis, y axis and a title.

```
import matplotlib.pyplot as plt
x = [1,2]
y = [3,4]
plt.plot(x, y)
plt.xlabel('X-axis Label')
plt.ylabel('Y-axis Label')
plt.title('Title of the Plot')
plt.show()
```

OUTPUT:

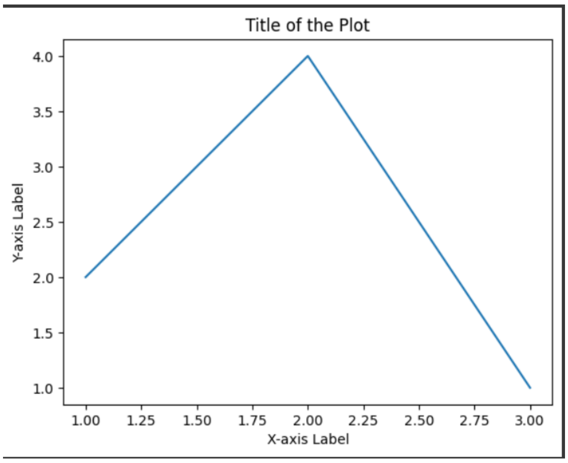


23.Write a Python program to draw a line using given axis values taken from a text file, with suitable label in the x axis, y axis and a title.

```
import matplotlib.pyplot as plt
x = [1,2,3]
```

```
y = [2,4,1]
plt.plot(x, y)
plt.xlabel('X-axis Label')
plt.ylabel('Y-axis Label')
plt.title('Title of the Plot')
plt.show()
```

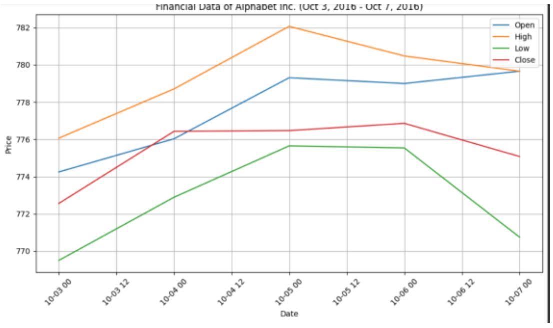
OUTPUT:



24.Write a Python program to draw line charts of the financial data of Alphabet Inc. between October 3, 2016 to October 7, 2016.

```
import pandas as pd
import matplotlib.pyplot as plt
financial_data = {
    'Date': ['10-03-16', '10-04-16', '10-05-16', '10-06-16', '10-07-16'],
    'Open': [774.25, 776.030029, 779.309998, 779, 779.659973],
    'High': [776.065002, 778.710022, 782.070007, 780.47998, 779.659973],
    'Low': [769.5, 772.890015, 775.650024, 775.539978, 770.75],
    'Close': [772.559998, 776.429993, 776.469971, 776.859985, 775.080017]
}
df = pd.DataFrame(financial_data)
df['Date'] = pd.to_datetime(df['Date'], format='%m-%d-%y')
plt.figure(figsize=(10, 6))
plt.plot(df['Date'], df['Open'], label='Open')
plt.plot(df['Date'], df['High'], label='High')
plt.plot(df['Date'], df['Low'], label='Low')
plt.plot(df['Date'], df['Close'], label='Close')
plt.xlabel('Date')
plt.ylabel('Price')
plt.title('Financial Data of Alphabet Inc. (Oct 3, 2016 - Oct 7, 2016)')
plt.xticks(rotation=45)
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```

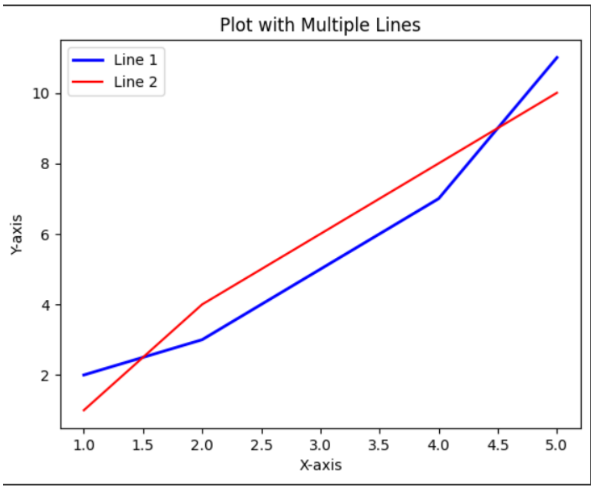
OUTPUT:



25.Write a Python program to plot two or more lines with legends, different widths and colors.

```
import matplotlib.pyplot as plt
x = [1, 2, 3, 4, 5]
y1 = [2, 3, 5, 7, 11]
y2 = [1, 4, 6, 8, 10]
plt.plot(x, y1, label='Line 1', color='blue', linewidth=2)
plt.plot(x, y2, label='Line 2', color='red', linewidth=1.5)
plt.legend()
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.title('Plot with Multiple Lines')
plt.show()
```

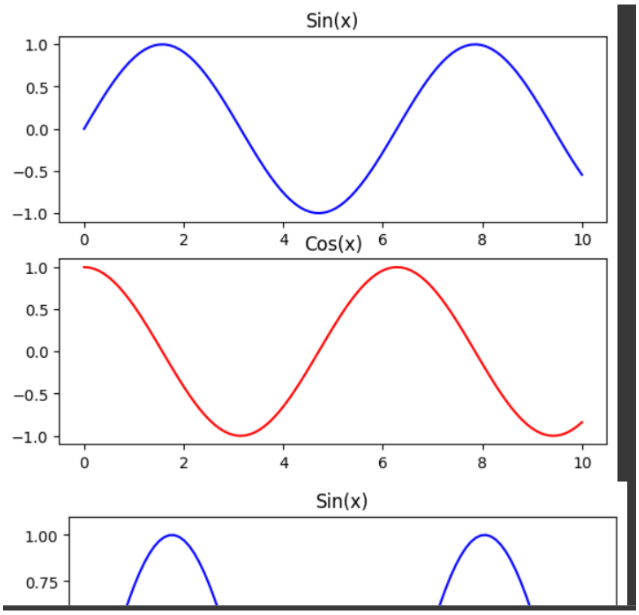
OUTPUT:



26. Write a Python program to create multiple plots.

```
import matplotlib.pyplot as plt
import numpy as np
x = np.linspace(0, 10, 100)
y1 = np.sin(x)
y2 = np.cos(x)
fig, axs = plt.subplots(2)
axs[0].plot(x, y1, color='blue')
axs[0].set_title('Sin(x)')
axs[1].plot(x, y2, color='red')
axs[1].set_title('Cos(x)')
plt.show()

plt.figure(1)
plt.plot(x, y1, color='blue')
plt.title('Sin(x)')
plt.show()

plt.figure(2)
plt.plot(x, y2, color='red')
plt.title('Cos(x)')
plt.show()
```

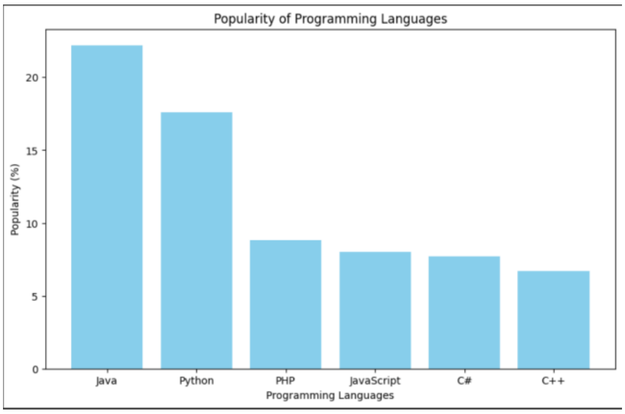
OUTPUT:



27. Write a Python programming to display a bar chart of the popularity of programming Languages.

```
import matplotlib.pyplot as plt
languages = ['Java', 'Python', 'PHP', 'JavaScript', 'C#', 'C++']
popularity = [22.2, 17.6, 8.8, 8, 7.7, 6.7]
plt.figure(figsize=(10, 6))
plt.bar(languages, popularity, color='skyblue')
plt.xlabel('Programming Languages')
plt.ylabel('Popularity (%)')
plt.title('Popularity of Programming Languages')
plt.show()
```

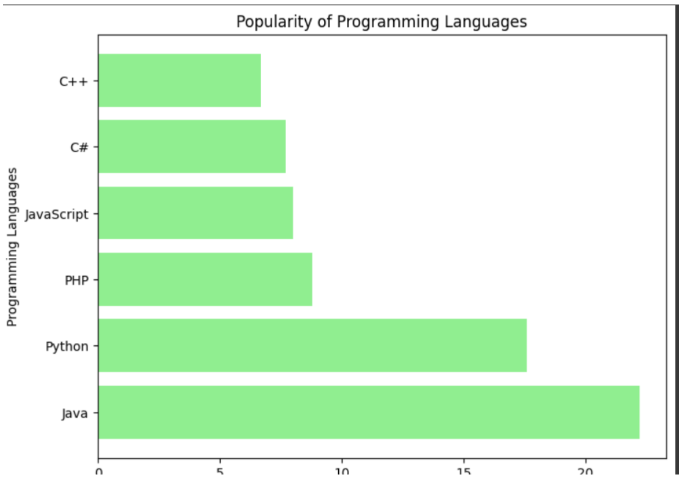
OUTPUT:



28.Write a Python programming to display a horizontal bar chart of the popularity of programming Languages.

```
import matplotlib.pyplot as plt
languages = ['Java', 'Python', 'PHP', 'JavaScript', 'C#', 'C++']
popularity = [22.2, 17.6, 8.8, 8, 7.7, 6.7]
plt.figure(figsize=(8, 6))
plt.barh(languages, popularity, color='lightgreen')
plt.xlabel('Popularity (%)')
plt.ylabel('Programming Languages')
plt.title('Popularity of Programming Languages')
plt.show()
```

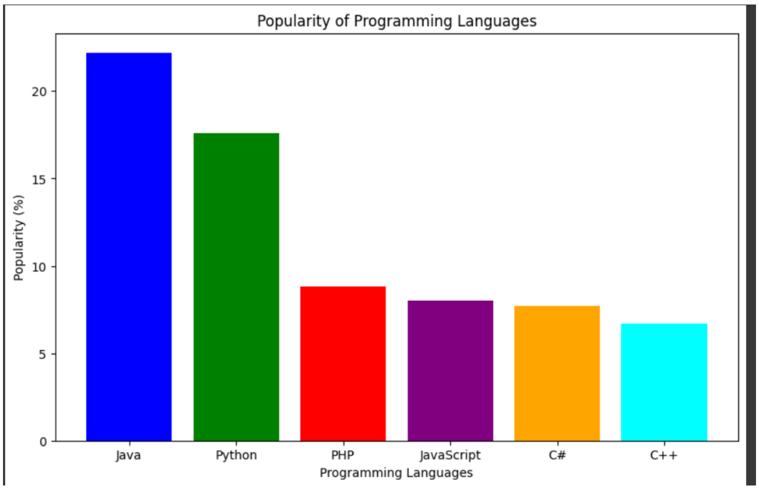
OUTPUT:



29.Write a Python programming to display a bar chart of the popularity of programming Languages. Use different color for each bar.

```
import matplotlib.pyplot as plt
languages = ['Java', 'Python', 'PHP', 'JavaScript', 'C#', 'C++']
popularity = [22.2, 17.6, 8.8, 8, 7.7, 6.7]
colors = ['blue', 'green', 'red', 'purple', 'orange', 'cyan']
plt.figure(figsize=(10, 6))
plt.bar(languages, popularity, color=colors)
plt.xlabel('Programming Languages')
plt.ylabel('Popularity (%)')
plt.title('Popularity of Programming Languages')
plt.show()
```

OUTPUT:

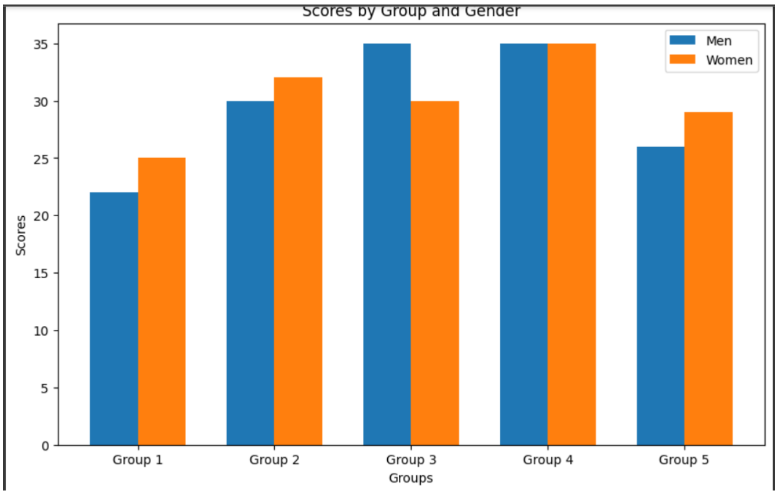


30.Write a Python program to create bar plot of scores by group and gender. Use multiple X values on the same chart for men and women.

```
import numpy as np
import matplotlib.pyplot as plt
men_means = (22, 30, 35, 35, 26)
women_means = (25, 32, 30, 35, 29)
```

```
group_labels = ['Group 1', 'Group 2', 'Group 3', 'Group 4', 'Group 5']
bar_width = 0.35
index = np.arange(len(group_labels))
plt.figure(figsize=(10, 6))
plt.bar(index, men_means, bar_width, label='Men')
plt.bar(index + bar_width, women_means, bar_width, label='Women')
plt.xlabel('Groups')
plt.ylabel('Scores')
plt.title('Scores by Group and Gender')
plt.xticks(index + bar_width / 2, group_labels)
plt.legend()
plt.show()
```

OUTPUT:

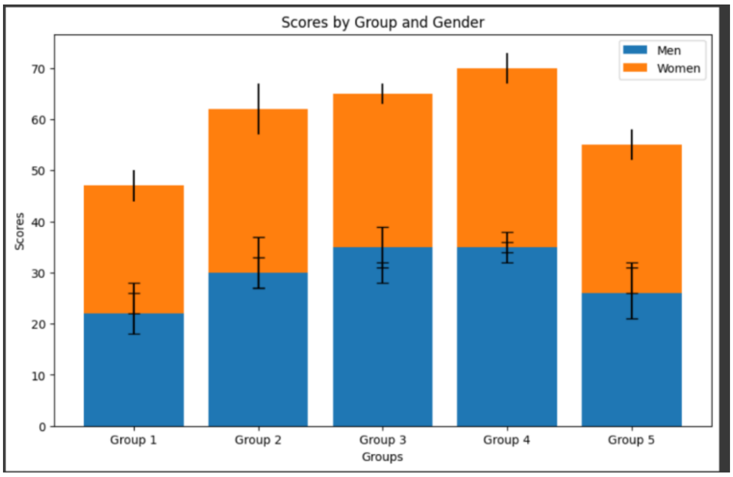


31.Write a Python program to create a stacked bar plot with error bars.

```
 import numpy as np
import matplotlib.pyplot as plt
men_means = (22, 30, 35, 35, 26)
women_means = (25, 32, 30, 35, 29)
men_std = (4, 3, 4, 1, 5)
women_std = (3, 5, 2, 3, 3)
```

```
group_labels = ['Group 1', 'Group 2', 'Group 3', 'Group 4', 'Group 5']

bottom_positions = np.array(men_means)

plt.figure(figsize=(10, 6))

bars1 = plt.bar(range(len(group_labels)), men_means, yerr=men_std, label='Men')

bars2 = plt.bar(range(len(group_labels)), women_means, yerr=women_std, bottom=bottom_positions, label='Women')

plt.xlabel('Groups')

plt.ylabel('Scores')

plt.title('Scores by Group and Gender')

plt.xticks(range(len(group_labels)), group_labels)

plt.legend()

plt.errorbar(range(len(group_labels)), men_means, yerr=men_std, fmt='none', ecolor='black', capsize=5)

plt.errorbar(range(len(group_labels)), women_means, yerr=women_std, fmt='none', ecolor='black', capsize=5)

plt.show()
```

OUTPUT:

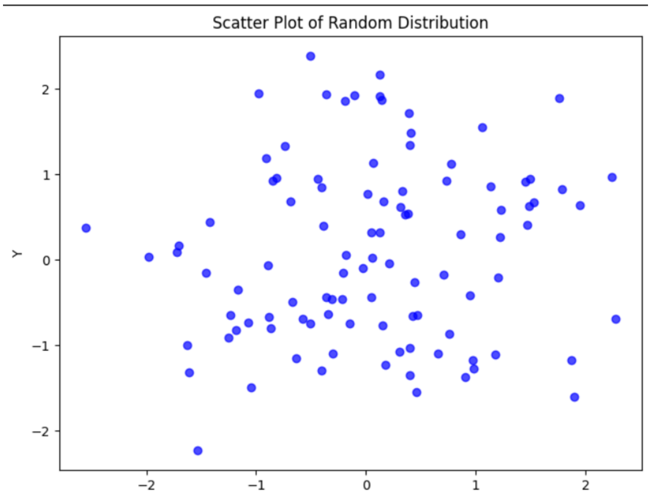


32.Write a Python program to draw a scatter graph taking a random distribution in X and Y and plotted against each other.

```
import numpy as np

import matplotlib.pyplot as plt

np.random.seed(0)
```

```
x = np.random.randn(100)
y = np.random.randn(100)
plt.figure(figsize=(8, 6))
plt.scatter(x, y, color='blue', alpha=0.7)
plt.xlabel('X')
plt.ylabel('Y')
plt.title('Scatter Plot of Random Distribution')
plt.show()
```

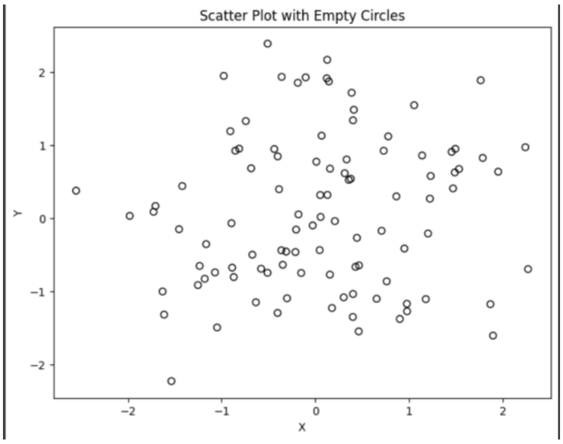
OUTPUT:



33.Write a Python program to draw a scatter plot with empty circles taking a random distribution in X and Y and plotted against each other.

```
import numpy as np
import matplotlib.pyplot as plt
np.random.seed(0)
x = np.random.randn(100)
y = np.random.randn(100)
plt.figure(figsize=(8, 6))
plt.scatter(x, y, color='blue', edgecolor='black', facecolor='none')
plt.xlabel('X')
plt.ylabel('Y')
plt.title('Scatter Plot with Empty Circles')
plt.show()
```

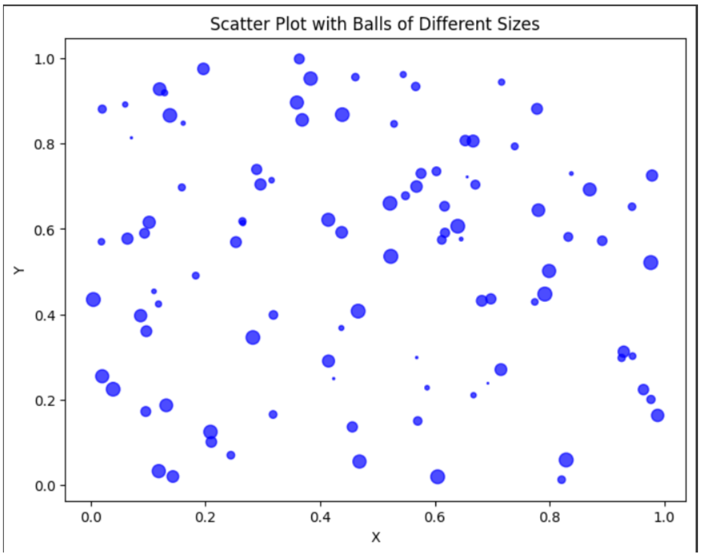
OUTPUT:



34.Write a Python program to draw a scatter plot using random distributions to generate balls of different sizes.

```
import numpy as np
import matplotlib.pyplot as plt
np.random.seed(0)
x = np.random.rand(100)
y = np.random.rand(100)
sizes = np.random.rand(100) * 100
plt.figure(figsize=(8, 6))
plt.scatter(x, y, s=sizes, color='blue', alpha=0.7)
plt.xlabel('X')
plt.ylabel('Y')
plt.title('Scatter Plot with Balls of Different Sizes')
plt.show()
```

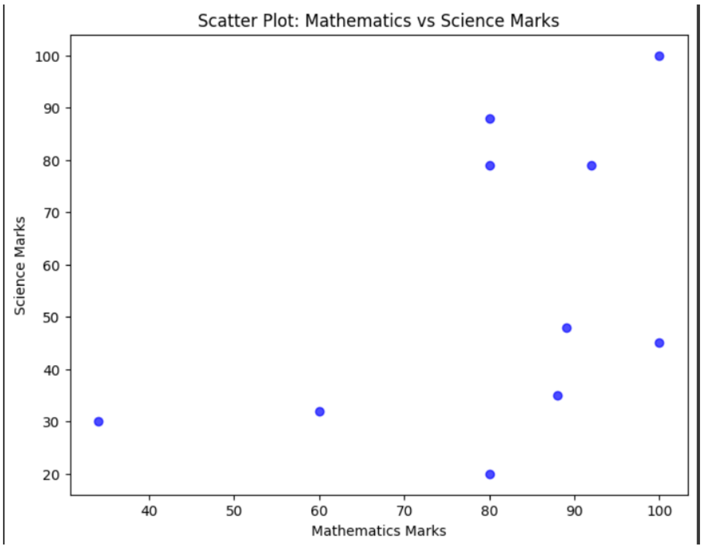
OUTPUT:



35.Write a Python program to draw a scatter plot comparing two subject marks of Mathematics and Science. Use marks of 10 students.

```
import matplotlib.pyplot as plt
math_marks = [88, 92, 80, 89, 100, 80, 60, 100, 80, 34]
science_marks = [35, 79, 79, 48, 100, 88, 32, 45, 20, 30]
marks_range = [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
plt.figure(figsize=(8, 6))
plt.scatter(math_marks, science_marks, color='blue', alpha=0.7)
plt.xlabel('Mathematics Marks')
plt.ylabel('Science Marks')
plt.title('Scatter Plot: Mathematics vs Science Marks')
plt.show()
```

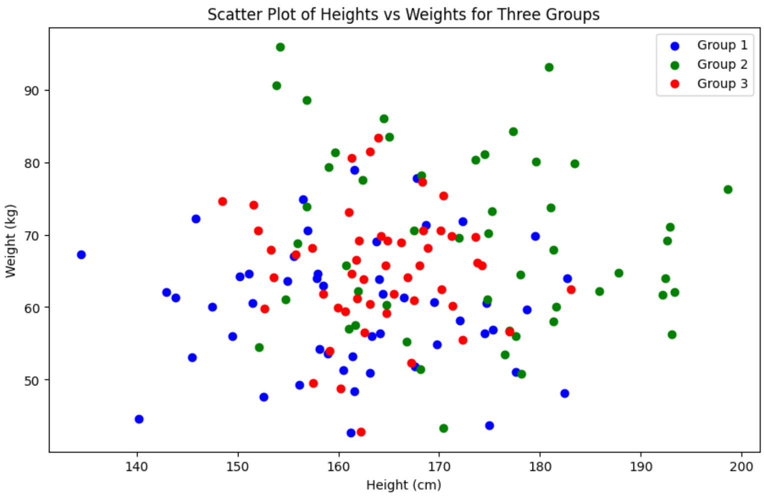
OUTPUT:



36.Write a Python program to draw a scatter plot for three different groups comparing weights and heights.

```
import matplotlib.pyplot as plt

import numpy as np

np.random.seed(0)

num_samples = 50

heights_group1 = np.random.normal(loc=160, scale=10, size=num_samples)

weights_group1 = np.random.normal(loc=60, scale=10, size=num_samples)

heights_group2 = np.random.normal(loc=170, scale=12, size=num_samples)

weights_group2 = np.random.normal(loc=70, scale=12, size=num_samples)

heights_group3 = np.random.normal(loc=165, scale=8, size=num_samples)

weights_group3 = np.random.normal(loc=65, scale=8, size=num_samples)

plt.figure(figsize=(10, 6))

plt.scatter(heights_group1, weights_group1, color='blue', label='Group 1')

plt.scatter(heights_group2, weights_group2, color='green', label='Group 2')

plt.scatter(heights_group3, weights_group3, color='red', label='Group 3')

plt.xlabel('Height (cm)')

plt.ylabel('Weight (kg)')

plt.title('Scatter Plot of Heights vs Weights for Three Groups')
plt.legend()
plt.show()
```

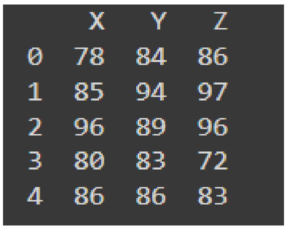
OUTPUT:



37.Write a Pandas program to create a dataframe from a dictionary and display it.

```
import pandas as pd
data = {'X': [78, 85, 96, 80, 86],
        'Y': [84, 94, 89, 83, 86],
        'Z': [86, 97, 96, 72, 83]}
df = pd.DataFrame(data)
print(df)
```

OUTPUT:

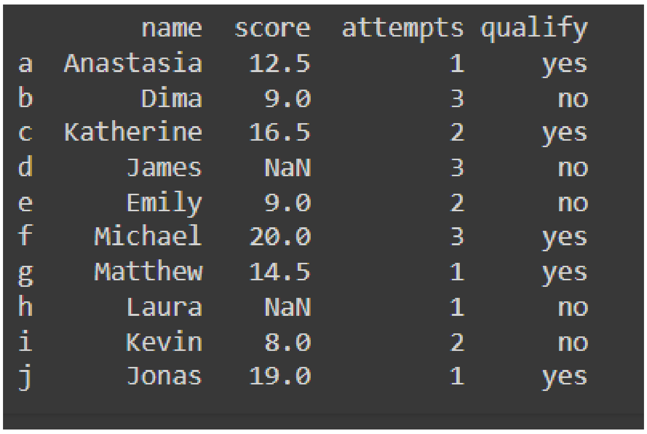


```
    X   Y   Z
0  78  84  86
1  85  94  97
2  96  89  96
3  80  83  72
4  86  86  83
```

38.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:;

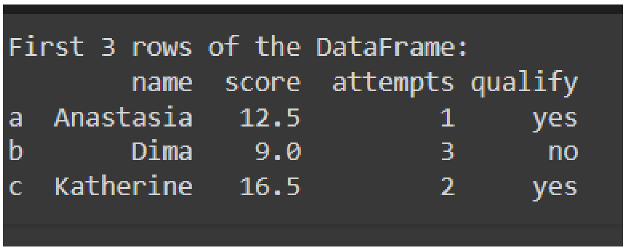


```
        name  score  attempts qualify
a  Anastasia   12.5         1     yes
b       Dima    9.0         3      no
c  Katherine   16.5         2     yes
d      James    NaN         3      no
e      Emily    9.0         2      no
f    Michael   20.0         3     yes
g    Matthew   14.5         1     yes
h      Laura    NaN         1      no
i      Kevin    8.0         2      no
j      Jonas   19.0         1     yes
```

39.Write a Pandas program to get the first 3 rows of a given 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)

first_3_rows = df.head(3)

print("First 3 rows of the DataFrame:")
print(first_3_rows)
```

OUTPUT:

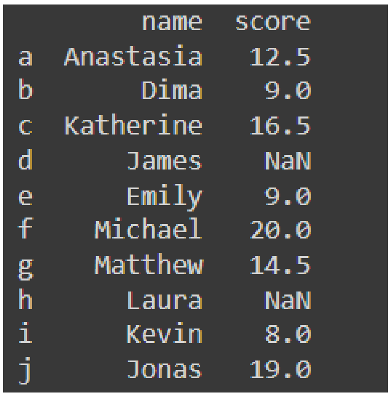


```
First 3 rows of the DataFrame:
        name  score  attempts qualify
a  Anastasia   12.5         1     yes
b       Dima    9.0         3      no
c  Katherine   16.5         2     yes
```

40. Write a Pandas program to select the name and score columns from the following 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)

selected_columns = df[['name', 'score']]

print(selected_columns)
```

OUTPUT:



```
        name  score
a  Anastasia   12.5
b       Dima    9.0
c  Katherine   16.5
d      James    NaN
e      Emily    9.0
f    Michael   20.0
g    Matthew   14.5
h      Laura    NaN
i      Kevin    8.0
j      Jonas   19.0
```