

## OUTPUT DOCUMENT – LAB EXPERIMENTS

**NAME :** Kovi Sai Ganesh

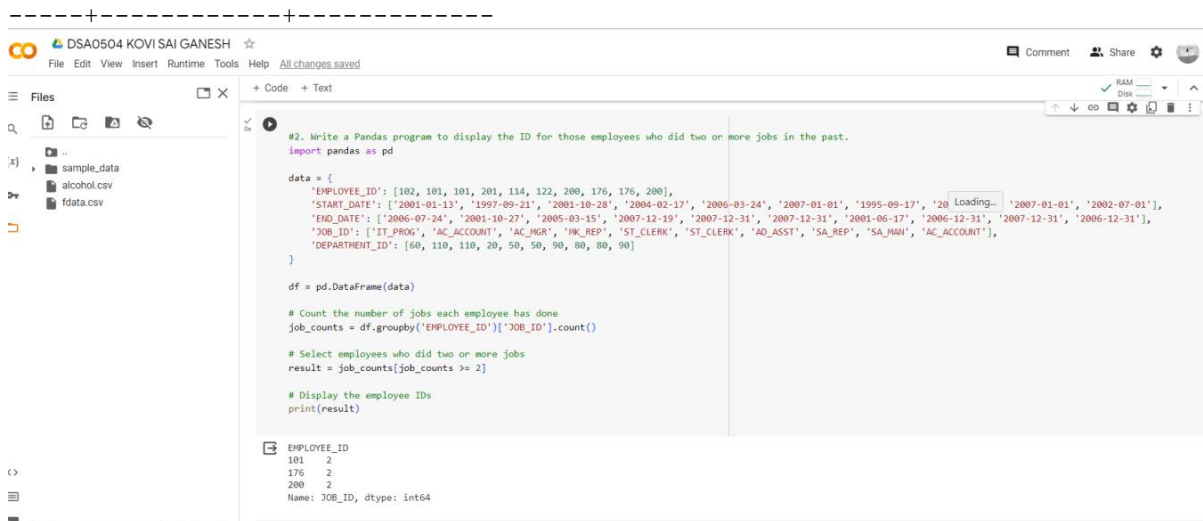
**REGISTER NUMBER :** 192124028

**COURSE CODE:** DSA0504

**COURSE NAME:** Query Processing for Data Science in Open Source Platform

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

```
+-----+-----+-----+-----+-----+
DEPARTMENT_ID | DEPARTMENT_NAME | MANAGER_ID | LOCATION_ID | +-----+
+-----+-----+-----+-----+-----+
Administration | 200 | 1700 | | 20 | Marketing | 201 | 1800 | | 30 |
Purchasing | 114 | 1700 | | 40 | Human Resources | 203 | 2400 | | 50 |
Shipping | 121 | 1500 | | 60 | IT | 103 | 1400 | | 70 | Public Relations
| 204 | 2700 | | 80 | Sales | 145 | 2500 | | 90 | Executive | 100 | 1700
| | 100 | Finance | 108 | 1700 | | 110 | Accounting | 205 | 1700 | |
120 | Treasury | 0 | 1700 | | 130 | Corporate Tax | 0 | 1700 | | 140 |
Control And Credit | 0 | 1700 | | 150 | Shareholder Services | 0 | 1700
| | 160 | Benefits | 0 | 1700 | | 170 | Manufacturing | 0 | 1700 | |
180 | Construction | 0 | 1700 | | 190 | Contracting | 0 | 1700 | | 200
| Operations | 0 | 1700 | | 210 | IT Support | 0 | 1700 | | 220 | NOC |
0 | 1700 | | 230 | IT Helpdesk | 0 | 1700 | | 240 | Government Sales |
0 | 1700 | | 250 | Retail Sales | 0 | 1700 | | 260 | Recruiting | 0 |
1700 | | 270 | Payroll | 0 | 1700 | +-----+-----+-----+-----+-----+
```



The screenshot shows a Jupyter Notebook with the following code and output:

```
#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', '2007-12-19', '2007-12-31', '2001-06-17', '2006-12-31', '2007-12-31', '2006-12-31'],
    '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]
}

df = pd.DataFrame(data)

# Count the number of jobs each employee has done
job_counts = df.groupby('EMPLOYEE_ID')['JOB_ID'].count()

# Select employees who did two or more jobs
result = job_counts[job_counts >= 2]

# Display the employee IDs
print(result)
```

The output shows the following data:

```
EMPLOYEE_ID
101      2
176      2
200      2
Name: JOB_ID, dtype: int64
```

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

```
+-----+-----+-----+-----+-----+
EMPLOYEE_ID | START_DATE | END_DATE | JOB_ID | DEPARTMENT_ID | +-----+
+-----+-----+-----+-----+-----+
01-13 | 2006-07-24 | IT_PROG | 60 | | 101 | 1997-09-21 | 2001-10-27 |
AC_ACCOUNT | 110 | | 101 | 2001-10-28 | 2005-03-15 | AC_MGR | 110 | | 201
| 2004-02-17 | 2007-12-19 | MK_REP | 20 | | 114 | 2006-03-24 | 2007-12-31
| ST_CLERK | 50 | | 122 | 2007-01-01 | 2007-12-31 | ST_CLERK | 50 | | 200
| 1995-09-17 | 2001-06-17 | AD_ASST | 90 | | 176 | 2006-03-24 | 2006-12-31
```

```
| SA_REP | 80 | | 176 | 2007-01-01 | 2007-12-31 | SA_MAN | 80 | | 200 |
2002-07-01 | 2006-12-31 | AC_ACCOUNT | 90 | +-----+-----+-----+
```

The screenshot shows a Jupyter Notebook with a file explorer on the left containing 'sample\_data', 'alcohol.csv', and 'fdata.csv'. The main area contains a code cell with the following Python code:

```
#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': [182, 181, 181, 201, 114, 122, 208, 176, 176, 208],
    'START_DATE': ['2001-01-13', '1997-09-21', '2001-10-28', '2004-02-17', '2006-03-24', '2007-01-01', '1995-09-17', '20 Loading--', '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]
}

df = pd.DataFrame(data)

# Count the number of jobs each employee has done
job_counts = df.groupby('EMPLOYEE_ID')['JOB_ID'].count()

# Select employees who did two or more jobs
result = job_counts[job_counts >= 2]

# Display the employee IDs
print(result)
```

The output shows the following result:

```
EMPLOYEE_ID
181      2
176      2
208      2
Name: JOB_ID, dtype: int64
```

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

```
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
JOB_ID | JOB_TITLE | MIN_SALARY | MAX_SALARY | +-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
AD PRES | President | 20080 | 40000 |
AD VP | Administration Vice President | 15000 | 30000 |
AD_ASST | Administration Assistant | 3000 | 6000 |
FI_MGR | Finance Manager | 8200 | 16000 |
FI_ACCOUNT | Accountant | 4200 | 9000 |
AC_MGR | Accounting Manager | 8200 | 16000 |
AC_ACCOUNT | Public Accountant | 4200 | 9000 |
SA_MAN | Sales Manager | 10000 | 20080 |
SA_REP | Sales Representative | 6000 | 12008 |
PU_MAN | Purchasing Manager | 8000 | 15000 |
PU_CLERK | Purchasing Clerk | 2500 | 5500 |
ST_MAN | Stock Manager | 5500 | 8500 |
ST_CLERK | Stock Clerk | 2008 | 5000 |
SH_CLERK | Shipping Clerk | 2500 | 5500 |
IT_PROG | Programmer | 4000 | 10000 |
MK_MAN | Marketing Manager | 9000 | 15000 |
MK_REP | Marketing Representative | 4000 | 9000 |
HR_REP | Human Resources Representative | 4000 | 9000 |
PR_REP | Public Relations Representative | 4500 | 10500 |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
```

The screenshot shows a Jupyter Notebook with a file explorer on the left containing 'sample\_data', 'alcohol.csv', and 'fdata.csv'. The main area contains a code cell with the following Python code:

```
df = pd.DataFrame(data)

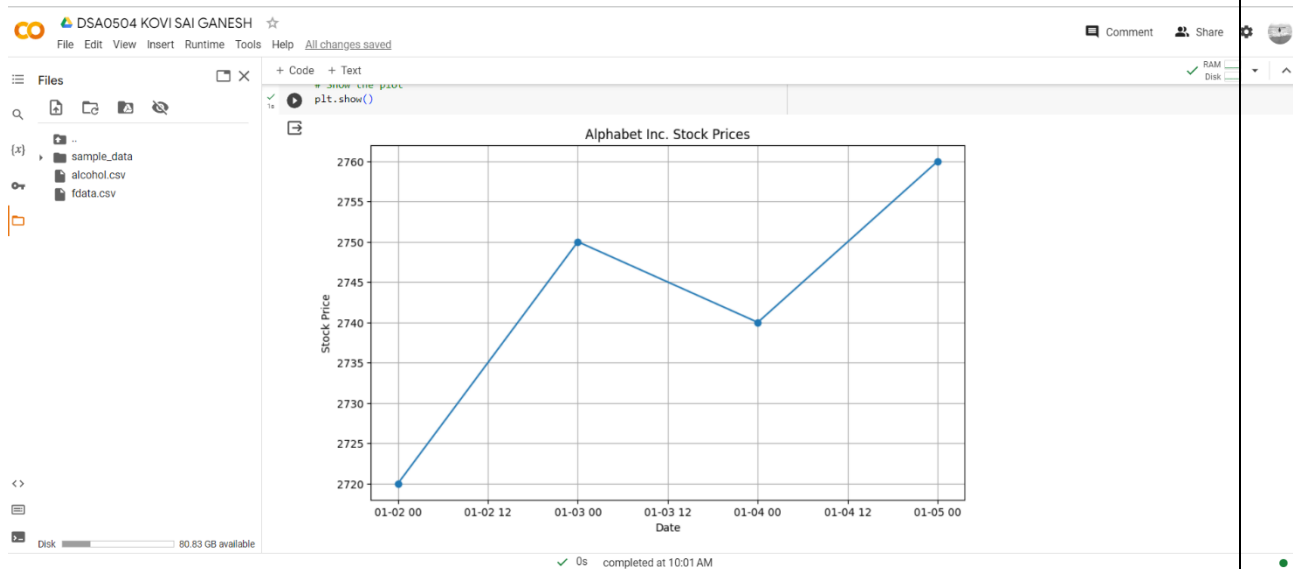
# Sort the DataFrame by job title in descending order
sorted_df = df.sort_values(by='JOB_TITLE', ascending=False)

# Display the result
print(sorted_df)
```

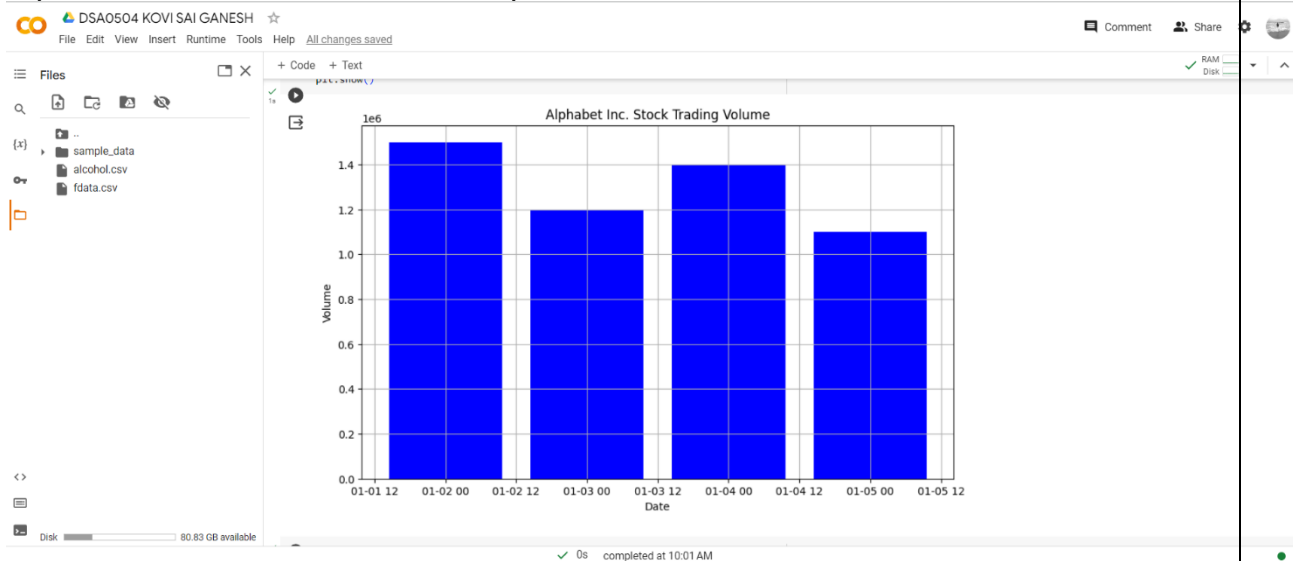
The output shows the following result:

```
   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.

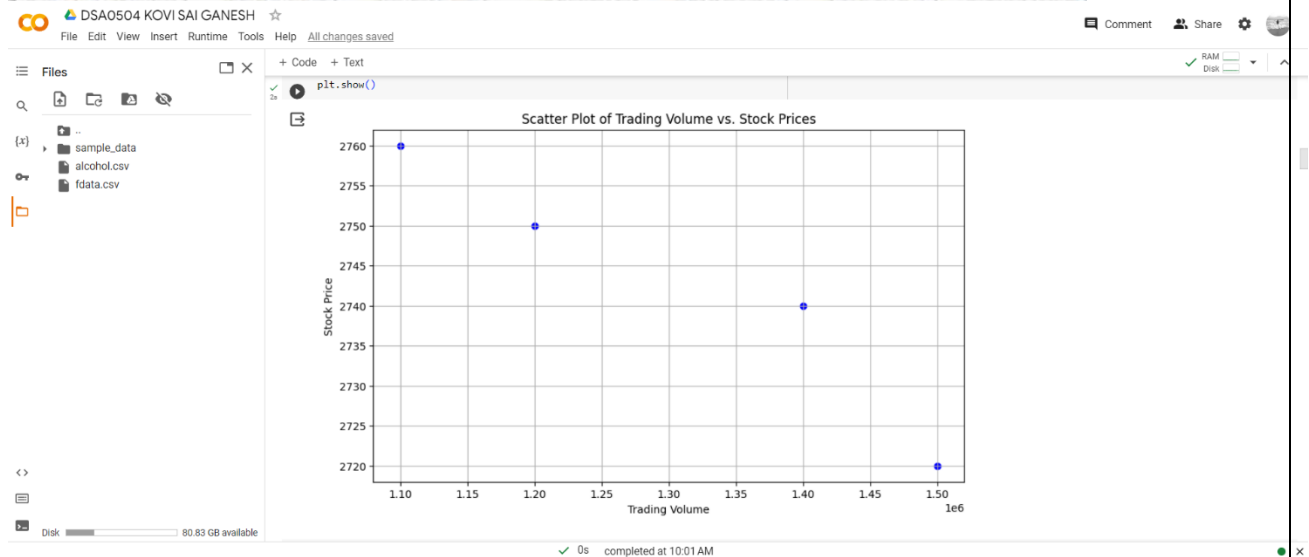


5. Write a Pandas program to create a bar plot of the trading volume of Alphabet Inc. stock between two specific dates.

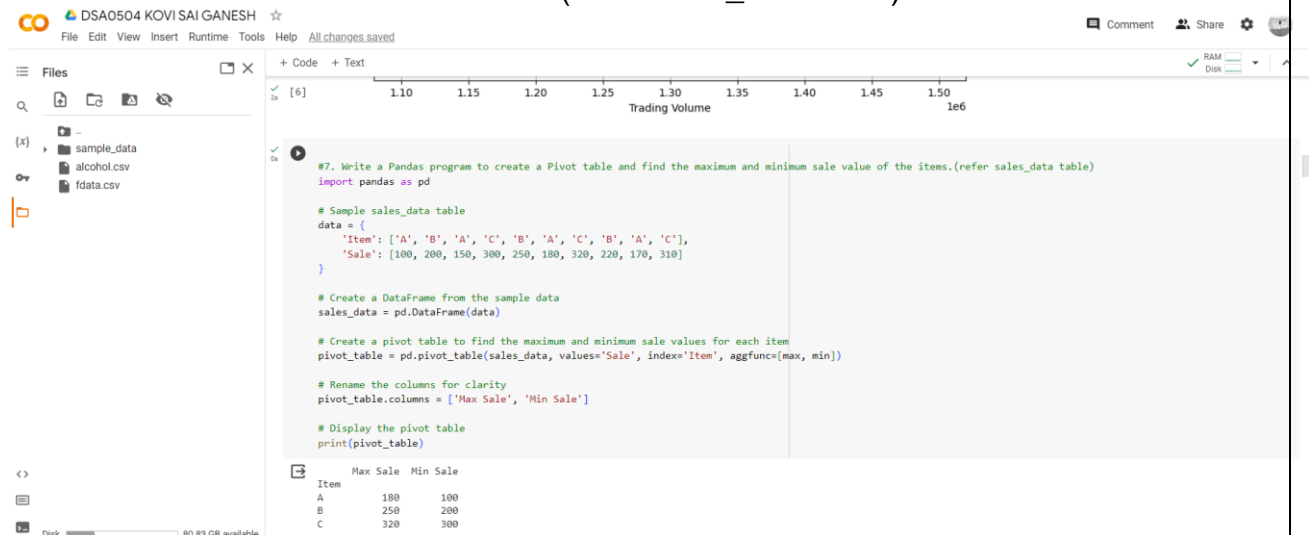


6. Write a Pandas program to create a scatter plot of the trading volume/stock prices of Alphabet Inc. stock between two specific dates.  
**alphabet\_stock\_data:**

| Date       | Open     | High     | Low      | Close   | Adj Close | Volume  |
|------------|----------|----------|----------|---------|-----------|---------|
| 01-04-2020 | 1122     | 1129.69  | 1097.45  | 1105.62 | 1105.62   | 2343100 |
| 02-04-2020 | 1098.26  | 1126.86  | 1096.4   | 1120.84 | 1120.84   | 1964900 |
| 03-04-2020 | 1119.015 | 1123.54  | 1079.81  | 1097.88 | 1097.88   | 2313400 |
| 06-04-2020 | 1138     | 1194.66  | 1130.94  | 1186.92 | 1186.92   | 2664700 |
| 07-04-2020 | 1221     | 1225     | 1182.23  | 1186.51 | 1186.51   | 2387300 |
| 08-04-2020 | 1206.5   | 1219.07  | 1188.16  | 1210.28 | 1210.28   | 1975100 |
| 09-04-2020 | 1224.08  | 1225.57  | 1196.735 | 1211.45 | 1211.45   | 2175400 |
| 13-04-2020 | 1209.18  | 1220.51  | 1187.598 | 1217.56 | 1217.56   | 1739800 |
| 14-04-2020 | 1245.09  | 1282.07  | 1236.93  | 1269.23 | 1269.23   | 2470400 |
| 15-04-2020 | 1245.61  | 1280.46  | 1240.4   | 1262.47 | 1262.47   | 1671700 |
| 16-04-2020 | 1274.1   | 1279     | 1242.62  | 1263.47 | 1263.47   | 2518100 |
| 17-04-2020 | 1284.85  | 1294.43  | 1271.23  | 1283.25 | 1283.25   | 1949000 |
| 20-04-2020 | 1271     | 1281.6   | 1261.37  | 1266.61 | 1266.61   | 1695500 |
| 21-04-2020 | 1247     | 1254.27  | 1209.71  | 1216.34 | 1216.34   | 2153000 |
| 22-04-2020 | 1245.54  | 1285.613 | 1242     | 1263.21 | 1263.21   | 2093100 |
| 23-04-2020 | 1271.55  | 1293.31  | 1265.67  | 1276.31 | 1276.31   | 1586200 |
| 24-04-2020 | 1261.17  | 1280.4   | 1249.45  | 1279.31 | 1279.31   | 1640400 |
| 27-04-2020 | 1296     | 1296.15  | 1269     | 1275.88 | 1275.88   | 1600600 |
| 28-04-2020 | 1287.93  | 1288.05  | 1232.2   | 1233.67 | 1233.67   | 2951300 |



7. Write a Pandas program to create a Pivot table and find the maximum and minimum sale value of the items.(refer sales\_data table)



8. Write a Pandas program to create a Pivot table and find the item wise unit sold. .(refer sales\_data table)

```

#8. Write a Pandas program to create a Pivot table and find the item wise unit sold. .(refer sales_data table)
import pandas as pd

# Sample sales_data table
data = {
    'Item': ['A', 'B', 'A', 'C', 'B', 'A', 'C', 'B', 'A', 'C'],
    'Unit Sold': [5, 10, 8, 12, 15, 9, 16, 11, 7, 13]
}

# Create a DataFrame from the sample data
sales_data = pd.DataFrame(data)

# Create a pivot table to find the total unit sold for each item
pivot_table = pd.pivot_table(sales_data, values='Unit Sold', index='Item', aggfunc='sum')

# Display the pivot table
print(pivot_table)

```

| Item | Unit Sold |
|------|-----------|
| A    | 29        |
| B    | 36        |
| C    | 41        |

9. Write a Pandas program to create a Pivot table and find the total sale amount region wise, manager wise, sales man wise. .(refer sales\_data table)

**Sales\_data:**

|  |  |  |      |  |  |
|--|--|--|------|--|--|
|  |  |  | Item |  |  |
|--|--|--|------|--|--|

**OrderDate Region Manager SalesMan Units Unit\_price Sale\_amt**

|      |        |  |    |  |          |
|------|--------|--|----|--|----------|
|      |        |  |    |  | 1,198.00 |
| East | Martha |  | 95 |  |          |

1-6-18 1,13,810.00

|  |           |  |              |    |        |
|--|-----------|--|--------------|----|--------|
|  | nn Shelli |  | Home Theater | 50 | 500.00 |
|--|-----------|--|--------------|----|--------|

1-23-18 25,000.00

|  |         |  |          |  |          |
|--|---------|--|----------|--|----------|
|  | nn Luis |  | Televisi |  | 1,198.00 |
|--|---------|--|----------|--|----------|

2-9-18 43,128.00

|  |         |  |            |    |        |
|--|---------|--|------------|----|--------|
|  | Timothy |  | Cell Phone | 27 | 225.00 |
|--|---------|--|------------|----|--------|

2-26-18 6,075.00

|      |         |
|------|---------|
| West | Timothy |
|------|---------|

|  |          |  |          |
|--|----------|--|----------|
|  | Televisi |  | 1,198.00 |
|--|----------|--|----------|

3-15-18 67,088.00

|      |        |           |              |    |        |
|------|--------|-----------|--------------|----|--------|
| East | Martha | Alexander | Home Theater | 60 | 500.00 |
|------|--------|-----------|--------------|----|--------|

4-1-18 30,000.00

|  |  |        |          |  |          |
|--|--|--------|----------|--|----------|
|  |  | Steven | Televisi |  | 1,198.00 |
|--|--|--------|----------|--|----------|

4-18-18 89,850.00

|  |         |  |          |  |          |
|--|---------|--|----------|--|----------|
|  | nn Luis |  | Televisi |  | 1,198.00 |
|--|---------|--|----------|--|----------|

5-5-18 1,07,820.00

|      |         |         |          |  |          |
|------|---------|---------|----------|--|----------|
| West | Douglas | Michael | Televisi |  | 1,198.00 |
|------|---------|---------|----------|--|----------|

5-22-18 38,336.00

|      |        |           |              |    |        |
|------|--------|-----------|--------------|----|--------|
| East | Martha | Alexander | Home Theater | 60 | 500.00 |
|------|--------|-----------|--------------|----|--------|

6-8-18 30,000.00

|  |          |  |          |  |          |
|--|----------|--|----------|--|----------|
|  | nn Sigal |  | Televisi |  | 1,198.00 |
|--|----------|--|----------|--|----------|

6-25-18 1,07,820.00

|      |        |       |              |    |        |
|------|--------|-------|--------------|----|--------|
| East | Martha | Diana | Home Theater | 29 | 500.00 |
|------|--------|-------|--------------|----|--------|

7-12-18 14,500.00

|      |         |
|------|---------|
| East | Douglas |
|------|---------|

|       |              |    |        |
|-------|--------------|----|--------|
| Karen | Home Theater | 81 | 500.00 |
|-------|--------------|----|--------|

7-29-18 40,500.00

|      |        |  |    |  |          |
|------|--------|--|----|--|----------|
| East | Martha |  | 35 |  | 1,198.00 |
|------|--------|--|----|--|----------|

8-15-18 41,930.00

|  |         |      |      |   |        |
|--|---------|------|------|---|--------|
|  | Douglas | John | Desk | 2 | 125.00 |
|--|---------|------|------|---|--------|

9-1-18 250.00

|      |        |           |             |    |       |
|------|--------|-----------|-------------|----|-------|
| East | Martha | Alexander | Video Games | 16 | 58.50 |
|------|--------|-----------|-------------|----|-------|

9-18-18 936.00

|  |           |  |              |    |        |
|--|-----------|--|--------------|----|--------|
|  | Ann Sigal |  | Home Theater | 28 | 500.00 |
|--|-----------|--|--------------|----|--------|

10-5-18 14,000.00

|      |        |           |            |    |        |
|------|--------|-----------|------------|----|--------|
| East | Martha | Alexander | Cell Phone | 64 | 225.00 |
|------|--------|-----------|------------|----|--------|

10-22-18 14,400.00

DSAO504 KOVI SAI GANESH

File Edit View Insert Runtime Tools Help All changes saved

Files

Up one level

sample\_data

alcohol.csv

fddata.csv

+ Code + Text

```

#8. Write a Pandas program to create a Pivot table and find the item wise unit sold. (refer sales_data table)
import pandas as pd

# Sample sales_data table
data = {
    'Item': ['A', 'B', 'A', 'C', 'B', 'A', 'C', 'B', 'A', 'C'],
    'Unit Sold': [5, 10, 8, 12, 15, 9, 16, 11, 7, 13]
}

# Create a DataFrame from the sample data
sales_data = pd.DataFrame(data)

# Create a pivot table to find the total unit sold for each item
pivot_table = pd.pivot_table(sales_data, values='Unit Sold', index='Item', aggfunc='sum')

# Display the pivot table
print(pivot_table)

```

Unit Sold  
Item      29  
A          36  
B          41

Comment Share

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.

**Expected Output:**

|   | A  | B        | C         | D           | E         |
|---|----|----------|-----------|-------------|-----------|
| 0 | 1  | 1.32921  | -0.770033 | -0.31628    | -0.99081  |
| 1 | 2  | -1.07082 | -1.43871  | 0.564417    | 0.295722  |
| 2 | 3  | -1.6264  | 0.219565  | 0.678805    | 1.88927   |
| 3 | 4  | 0.961538 | 0.104011  | -0.481165   | 0.850229  |
| 4 | 5  | 1.45342  | 1.05774   | 0.165562    | 0.515018  |
| 5 | 6  | -1.33694 | 0.562861  | 1.39285     | -0.063328 |
| 6 | 7  | 0.121668 | 1.2076    | -0.00204021 | 1.6278    |
| 7 | 8  | 0.354493 | 1.03753   | -0.385684   | 0.519818  |
| 8 | 9  | 1.68658  | -1.32596  | 1.42898     | -2.08935  |
| 9 | 10 | -0.12982 | 0.631523  | -0.586538   | 0.29072   |

```

df = pd.DataFrame(data, columns=['Column1', 'Column2', 'Column3', 'Column4'])
styled_df = df.style.applymap(lambda val: f'color: ("red" if val < 0 else "black")')
# Display the styled DataFrame
styled_df

```

|   | Column1   | Column2   | Column3   | Column4   |
|---|-----------|-----------|-----------|-----------|
| 0 | -0.036970 | -0.288625 | -0.550897 | 1.050166  |
| 1 | 0.519324  | -0.658154 | 0.321001  | 0.141010  |
| 2 | -0.320677 | -0.283296 | -1.727262 | 2.394471  |
| 3 | 0.862895  | 0.101486  | -2.237401 | 0.594556  |
| 4 | -0.752640 | 0.239556  | 1.189702  | 0.303059  |
| 5 | -0.447863 | -1.781747 | 0.734880  | -1.256585 |
| 6 | 0.633450  | 0.267513  | -0.833781 | 1.621736  |
| 7 | -0.340908 | 1.520432  | 0.239833  | 1.223547  |
| 8 | 0.107300  | 1.009721  | -0.701498 | 0.595555  |
| 9 | 0.525514  | -1.253191 | -1.062624 | -0.624492 |

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.

|   | A  | B        | C        | D           | E         |
|---|----|----------|----------|-------------|-----------|
| 0 | 1  | 1.32921  | nan      | -0.31628    | -0.99081  |
| 1 | 2  | -1.07082 | -1.43871 | 0.564417    | 0.295722  |
| 2 | 3  | -1.6264  | 0.219565 | 0.678805    | 1.88927   |
| 3 | 4  | 0.961538 | 0.104011 | nan         | 0.850229  |
| 4 | 5  | nan      | 1.05774  | 0.165562    | 0.515018  |
| 5 | 6  | -1.33694 | 0.562861 | 1.39285     | -0.063328 |
| 6 | 7  | 0.121668 | 1.2076   | -0.00204021 | 1.6278    |
| 7 | 8  | 0.354493 | 1.03753  | -0.385684   | 0.519818  |
| 8 | 9  | 1.68658  | -1.32596 | 1.42898     | -2.08935  |
| 9 | 10 | -0.12982 | 0.631523 | -0.586538   | nan       |



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  | 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       |

Negative numbers red and positive numbers black:  
 <ipython-input-12-0152cf84b42a>:21: FutureWarning: `null\_color` is deprecated: use `color` instead  
 df.style.highlight\_null(null\_color='red')

|   | A         | B         | C         | D         | E         |
|---|-----------|-----------|-----------|-----------|-----------|
| 0 | 1.000000  | 1.329212  | nan       | -0.316280 | -0.990810 |
| 1 | 2.000000  | -1.070816 | -1.438713 | 0.564417  | 0.295722  |
| 2 | 3.000000  | -1.626404 | 0.219565  | 0.678805  | 1.889273  |
| 3 | 4.000000  | 0.961538  | 0.104011  | nan       | 0.850229  |
| 4 | 5.000000  | nan       | 1.057737  | 0.165562  | 0.515018  |
| 5 | 6.000000  | -1.336936 | 0.562861  | 1.392855  | -0.063328 |
| 6 | 7.000000  | 0.121668  | 1.207603  | -0.002040 | 1.627796  |
| 7 | 8.000000  | 0.354493  | 1.037528  | -0.385684 | 0.519818  |
| 8 | 9.000000  | 1.686583  | -1.325963 | 1.428984  | -2.089354 |
| 9 | 10.000000 | -0.129820 | 0.631523  | -0.586538 | 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.

|   | A  | B        | C        | D           | E         |
|---|----|----------|----------|-------------|-----------|
| 0 | 1  | 1.32921  | nan      | -0.31628    | -0.99081  |
| 1 | 2  | -1.07082 | -1.43871 | 0.564417    | 0.295722  |
| 2 | 3  | -1.6264  | 0.219565 | 0.678805    | 1.88927   |
| 3 | 4  | 0.961538 | 0.104011 | nan         | 0.850229  |
| 4 | 5  | nan      | 1.05774  | 0.165562    | 0.515018  |
| 5 | 6  | -1.33694 | 0.562861 | 1.39285     | -0.063328 |
| 6 | 7  | 0.121668 | 1.2076   | -0.00204021 | 1.6278    |
| 7 | 8  | 0.354493 | 1.03753  | -0.385684   | 0.519818  |
| 8 | 9  | 1.68658  | -1.32596 | 1.42898     | -2.08935  |
| 9 | 10 | -0.12982 | 0.631523 | -0.586538   | nan       |

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  | 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       |

Background:black - fontcolor:yellow

|   | A         | B         | C         | D         | E         |
|---|-----------|-----------|-----------|-----------|-----------|
| 0 | 1.000000  | 1.329212  | nan       | -0.316280 | -0.990810 |
| 1 | 2.000000  | -1.070816 | -1.438713 | 0.564417  | 0.295722  |
| 2 | 3.000000  | -1.626404 | 0.219565  | 0.678805  | 1.889273  |
| 3 | 4.000000  | 0.961538  | 0.104011  | nan       | 0.850229  |
| 4 | 5.000000  | nan       | 1.057737  | 0.165562  | 0.515018  |
| 5 | 6.000000  | -1.336936 | 0.562861  | 1.392855  | -0.063328 |
| 6 | 7.000000  | 0.121668  | 1.207603  | -0.002040 | 1.627796  |
| 7 | 8.000000  | 0.354493  | 1.037528  | -0.385684 | 0.519818  |
| 8 | 9.000000  | 1.686583  | -1.325963 | 1.428984  | -2.089354 |
| 9 | 10.000000 | -0.129820 | 0.631523  | -0.586538 | nan       |

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

|    | ord_no  | purch_amt | ord_date   | customer_id | salesman_id |
|----|---------|-----------|------------|-------------|-------------|
| 0  | 70001.0 | 150.50    | 2012-10-05 | 3002        | 5002.0      |
| 1  | NaN     | 270.65    | 2012-09-10 | 3001        | 5003.0      |
| 2  | 70002.0 | 65.26     | NaN        | 3001        | 5001.0      |
| 3  | 70004.0 | 110.50    | 2012-08-17 | 3003        | NaN         |
| 4  | NaN     | 948.50    | 2012-09-10 | 3002        | 5002.0      |
| 5  | 70005.0 | 2400.60   | 2012-07-27 | 3001        | 5001.0      |
| 6  | NaN     | 5760.00   | 2012-09-10 | 3001        | 5001.0      |
| 7  | 70010.0 | 1983.43   | 2012-10-10 | 3004        | NaN         |
| 8  | 70003.0 | 2480.40   | 2012-10-10 | 3003        | 5003.0      |
| 9  | 70012.0 | 250.45    | 2012-06-27 | 3002        | 5002.0      |
| 10 | NaN     | 75.29     | 2012-08-17 | 3001        | 5003.0      |
| 11 | 70013.0 | 3045.60   | 2012-04-25 | 3001        | NaN         |

➡ Original Orders DataFrame:

|    | ord_no  | purch_amt | ord_date   | customer_id | salesman_id |
|----|---------|-----------|------------|-------------|-------------|
| 0  | 70001.0 | 150.50    | 2012-10-05 | 3002        | 5002.0      |
| 1  | NaN     | 270.65    | 2012-09-10 | 3001        | 5003.0      |
| 2  | 70002.0 | 65.26     | NaN        | 3001        | 5001.0      |
| 3  | 70004.0 | 110.50    | 2012-08-17 | 3003        | NaN         |
| 4  | NaN     | 948.50    | 2012-09-10 | 3002        | 5002.0      |
| 5  | 70005.0 | 2400.60   | 2012-07-27 | 3001        | 5001.0      |
| 6  | NaN     | 5760.00   | 2012-09-10 | 3001        | 5001.0      |
| 7  | 70010.0 | 1983.43   | 2012-10-10 | 3004        | NaN         |
| 8  | 70003.0 | 2480.40   | 2012-10-10 | 3003        | 5003.0      |
| 9  | 70012.0 | 250.45    | 2012-06-27 | 3002        | 5002.0      |
| 10 | NaN     | 75.29     | 2012-08-17 | 3001        | 5003.0      |
| 11 | 70013.0 | 3045.60   | 2012-04-25 | 3001        | NaN         |

Missing values of the said dataframe:

|    | ord_no | purch_amt | ord_date | customer_id | salesman_id |
|----|--------|-----------|----------|-------------|-------------|
| 0  | False  | False     | False    | False       | False       |
| 1  | True   | False     | False    | False       | False       |
| 2  | False  | False     | True     | False       | False       |
| 3  | False  | False     | 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       |
| 11 | False  | False     | False    | False       | True        |

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

|    | ord_no | purch_amt | ord_date   | customer_id | salesman_id |
|----|--------|-----------|------------|-------------|-------------|
| 0  | 70001  | 150.5     | ?          | 3002        | 5002        |
| 1  | NaN    | 270.65    | 2012-09-10 | 3001        | 5003        |
| 2  | 70002  | 65.26     | NaN        | 3001        | ?           |
| 3  | 70004  | 110.5     | 2012-08-17 | 3003        | 5001        |
| 4  | NaN    | 948.5     | 2012-09-10 | 3002        | NaN         |
| 5  | 70005  | 2400.6    | 2012-07-27 | 3001        | 5002        |
| 6  | --     | 5760      | 2012-09-10 | 3001        | 5001        |
| 7  | 70010  | ?         | 2012-10-10 | 3004        | ?           |
| 8  | 70003  | 12.43     | 2012-10-10 | --          | 5003        |
| 9  | 70012  | 2480.4    | 2012-06-27 | 3002        | 5002        |
| 10 | NaN    | 250.45    | 2012-08-17 | 3001        | 5003        |
| 11 | 70013  | 3045.6    | 2012-04-25 | 3001        | --          |

➡ Original Orders DataFrame:

|    | ord_no | purch_amt | ord_date   | customer_id | salesman_id |
|----|--------|-----------|------------|-------------|-------------|
| 0  | 70001  | 150.5     | ?          | 3002        | 5002        |
| 1  | NaN    | 270.65    | 2012-09-10 | 3001        | 5003        |
| 2  | 70002  | 65.26     | NaN        | 3001        | ?           |
| 3  | 70004  | 110.5     | 2012-08-17 | 3003        | 5001        |
| 4  | NaN    | 948.5     | 2012-09-10 | 3002        | NaN         |
| 5  | 70005  | 2400.6    | 2012-07-27 | 3001        | 5002        |
| 6  | --     | 5760      | 2012-09-10 | 3001        | 5001        |
| 7  | 70010  | ?         | 2012-10-10 | 3004        | ?           |
| 8  | 70003  | 12.43     | 2012-10-10 | --          | 5003        |
| 9  | 70012  | 2480.4    | 2012-06-27 | 3002        | 5002        |
| 10 | NaN    | 250.45    | 2012-08-17 | 3001        | 5003        |
| 11 | 70013  | 3045.6    | 2012-04-25 | 3001        | --          |

Replace the missing values with NaN:

|    | ord_no  | purch_amt | ord_date   | customer_id | salesman_id |
|----|---------|-----------|------------|-------------|-------------|
| 0  | 70001.0 | 150.50    | NaN        | 3002.0      | 5002.0      |
| 1  | NaN     | 270.65    | 2012-09-10 | 3001.0      | 5003.0      |
| 2  | 70002.0 | 65.26     | NaN        | 3001.0      | NaN         |
| 3  | 70004.0 | 110.50    | 2012-08-17 | 3003.0      | 5001.0      |
| 4  | NaN     | 948.50    | 2012-09-10 | 3002.0      | NaN         |
| 5  | 70005.0 | 2400.60   | 2012-07-27 | 3001.0      | 5002.0      |
| 6  | NaN     | 5760.00   | 2012-09-10 | 3001.0      | 5001.0      |
| 7  | 70010.0 | NaN       | 2012-10-10 | 3004.0      | NaN         |
| 8  | 70003.0 | 12.43     | 2012-10-10 | NaN         | 5003.0      |
| 9  | 70012.0 | 2480.40   | 2012-06-27 | 3002.0      | 5002.0      |
| 10 | NaN     | 250.45    | 2012-08-17 | 3001.0      | 5003.0      |
| 11 | 70013.0 | 3045.60   | 2012-04-25 | 3001.0      | NaN         |

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

|    | ord_no  | purch_amt | ord_date   | customer_id |
|----|---------|-----------|------------|-------------|
| 0  | NaN     | NaN       | NaN        | NaN         |
| 1  | NaN     | 270.65    | 2012-09-10 | 3001.0      |
| 2  | 70002.0 | 65.26     | NaN        | 3001.0      |
| 3  | NaN     | NaN       | NaN        | NaN         |
| 4  | NaN     | 948.50    | 2012-09-10 | 3002.0      |
| 5  | 70005.0 | 2400.60   | 2012-07-27 | 3001.0      |
| 6  | NaN     | 5760.00   | 2012-09-10 | 3001.0      |
| 7  | 70010.0 | 1983.43   | 2012-10-10 | 3004.0      |
| 8  | 70003.0 | 2480.40   | 2012-10-10 | 3003.0      |
| 9  | 70012.0 | 250.45    | 2012-06-27 | 3002.0      |
| 10 | NaN     | 75.29     | 2012-08-17 | 3001.0      |
| 11 | NaN     | NaN       | NaN        | NaN         |

```
Original Orders DataFrame:
   ord_no  purch_amt  ord_date  customer_id
0      NaN         NaN         NaN         NaN
1      NaN         270.65  2012-09-10      3001.0
2    70002.0         65.26         NaN      3001.0
3      NaN         NaN         NaN         NaN
4      NaN         948.50  2012-09-10      3002.0
5    70005.0        2400.60  2012-07-27      3001.0
6      NaN        5760.00  2012-09-10      3001.0
7    70010.0        1983.43  2012-10-10      3004.0
8    70003.0        2480.40  2012-10-10      3003.0
9    70012.0         250.45  2012-06-27      3002.0
10     NaN         75.29  2012-08-17      3001.0
11     NaN         NaN         NaN         NaN
```

Keep the rows with at least 2 NaN values of the said DataFrame:

```
   ord_no  purch_amt  ord_date  customer_id
1      NaN         270.65  2012-09-10      3001.0
2    70002.0         65.26         NaN      3001.0
4      NaN         948.50  2012-09-10      3002.0
5    70005.0        2400.60  2012-07-27      3001.0
6      NaN        5760.00  2012-09-10      3001.0
7    70010.0        1983.43  2012-10-10      3004.0
8    70003.0        2480.40  2012-10-10      3003.0
9    70012.0         250.45  2012-06-27      3002.0
10     NaN         75.29  2012-08-17      3001.0
```

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

```
   school class  name date_of_birth  age  height  weight  address
S1   s001     V  Alberto Franco   15/05/2002   12   173    35  street1
S2   s002     V   Gino Mcneill   17/05/2002   12   192    32  street2
S3   s003     VI   Ryan Parkes   16/02/1999   13   186    33  street3
S4   s001     VI   Eesha Hinton   25/09/1998   13   167    30  street1
S5   s002     V   Gino Mcneill   11/05/2002   14   151    31  street2
S6   s004     VI   David Parkes   15/09/1997   12   159    32  street4
```

```
school_code class  name date_of_birth  age  height  weight \
S1   s001     V  Alberto Franco   15/05/2002   12   173    35
S2   s002     V   Gino Mcneill   17/05/2002   12   192    32
S3   s003     VI   Ryan Parkes   16/02/1999   13   186    33
S4   s001     VI   Eesha Hinton   25/09/1998   13   167    30
S5   s002     V   Gino Mcneill   11/05/2002   14   151    31
S6   s004     VI   David Parkes   15/09/1997   12   159    32
```

```
address
S1  street1
S2  street2
S3  street3
S4  street1
S5  street2
S6  street4
```

Split the data on school\_code wise:

Group:

s001

```
school_code class  name date_of_birth  age  height  weight \
S1   s001     V  Alberto Franco   15/05/2002   12   173    35
S4   s001     VI   Eesha Hinton   25/09/1998   13   167    30
```

```
address
S1  street1
S4  street1
```

Type of the object:

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

Group:

s002

```
school_code class  name date_of_birth  age  height  weight  address
S2   s002     V   Gino Mcneill   17/05/2002   12   192    32  street2
S5   s002     V   Gino Mcneill   11/05/2002   14   151    31  street2
```

Type of the object:

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

✓ 0s completed at 10:01 AM

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.

|    | school | class | name           | date_Of_Birth | age | height | weight | address |
|----|--------|-------|----------------|---------------|-----|--------|--------|---------|
| S1 | s001   | V     | Alberto Franco | 15/05/2002    | 12  | 173    | 35     | street1 |
| S2 | s002   | V     | Gino Mcneill   | 17/05/2002    | 12  | 192    | 32     | street2 |
| S3 | s003   | VI    | Ryan Parkes    | 16/02/1999    | 13  | 186    | 33     | street3 |
| S4 | s001   | VI    | Eesha Hinton   | 25/09/1998    | 13  | 167    | 30     | street1 |

Original DataFrame:

|    | school_code | class | name           | date_of_birth | age | height | weight | \ |
|----|-------------|-------|----------------|---------------|-----|--------|--------|---|
| S1 | s001        | V     | Alberto Franco | 15/05/2002    | 12  | 173    | 35     |   |
| S2 | s002        | V     | Gino Mcneill   | 17/05/2002    | 12  | 192    | 32     |   |
| S3 | s003        | VI    | Ryan Parkes    | 16/02/1999    | 13  | 186    | 33     |   |
| S4 | s001        | VI    | Eesha Hinton   | 25/09/1998    | 13  | 167    | 30     |   |
| S5 | s002        | V     | Gino Mcneill   | 11/05/2002    | 14  | 151    | 31     |   |
| S6 | s004        | VI    | David Parkes   | 15/09/1997    | 12  | 159    | 32     |   |

|    | address |
|----|---------|
| S1 | street1 |
| S2 | street2 |
| S3 | street3 |
| S4 | street1 |
| S5 | street2 |
| S6 | street4 |

Mean, min, and max value of age for each school with customized column names:

|             | Age_Mean | Age_Max | Age_Min |
|-------------|----------|---------|---------|
| school_code |          |         |         |
| s001        | 12.5     | 13      | 12      |
| s002        | 13.0     | 14      | 12      |
| s003        | 13.0     | 13      | 13      |
| s004        | 12.0     | 12      | 12      |

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

|    | school | class | name           | date_Of_Birth | age | height | weight | address |
|----|--------|-------|----------------|---------------|-----|--------|--------|---------|
| S1 | s001   | V     | Alberto Franco | 15/05/2002    | 12  | 173    | 35     | street1 |
| S2 | s002   | V     | Gino Mcneill   | 17/05/2002    | 12  | 192    | 32     | street2 |
| S3 | s003   | VI    | Ryan Parkes    | 16/02/1999    | 13  | 186    | 33     | street3 |
| S4 | s001   | VI    | Eesha Hinton   | 25/09/1998    | 13  | 167    | 30     | street1 |
| S5 | s002   | V     | Gino Mcneill   | 11/05/2002    | 14  | 151    | 31     | street2 |
| S6 | s004   | VI    | David Parkes   | 15/09/1997    | 12  | 159    | 32     | street4 |

```

Original DataFrame:
  school_code class  name date_of_birth age height weight \
S1    s001    V  Alberto Franco   15/05/2002  12   173   35
S2    s002    V    Gino McNeill  17/05/2002  12   192   32
S3    s003    VI   Ryan Parkes   16/02/1999  13   186   33
S4    s001    VI   Eesha Hinton  25/09/1998  13   167   30
S5    s002    V    Gino McNeill  11/05/2002  14   151   31
S6    s004    VI   David Parkes   15/09/1997  12   159   32

address
S1 street1
S2 street2
S3 street3
S4 street1
S5 street2
S6 street4

Split the said data on school_code wise:

Group:
s001
  school_code class  name date_of_birth age height weight \
S1    s001    V  Alberto Franco   15/05/2002  12   173   35
S4    s001    VI   Eesha Hinton  25/09/1998  13   167   30

address
S1 street1
S4 street1

Type of the object:
<class 'pandas.core.groupby.generic.DataFrameGroupBy'>

Group:
s002
  school_code class  name date_of_birth age height weight address
S2    s002    V    Gino McNeill  17/05/2002  12   192   32 street2
S5    s002    V    Gino McNeill  11/05/2002  14   151   31 street2

Type of the object:
<class 'pandas.core.groupby.generic.DataFrameGroupBy'>

Group:
s003
  school_code class  name date_of_birth age height weight address
S3    s003    VI   Ryan Parkes   16/02/1999  13   186   33 street3

Type of the object:
<class 'pandas.core.groupby.generic.DataFrameGroupBy'>

Group:
s004
  school_code class  name date_of_birth age height weight address
S6    s004    VI   David Parkes   15/09/1997  12   159   32 street4

Type of the object:
<class 'pandas.core.groupby.generic.DataFrameGroupBy'>

```

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.

|   | Year | WHO region      | Country               | Beverage Types | Display Value |
|---|------|-----------------|-----------------------|----------------|---------------|
| 3 | 1986 | Western Pacific | Viet Nam              | Wine           | 0.00          |
| 1 | 1986 | Americas        | Uruguay               | Other          | 0.50          |
| 2 | 1985 | Africa          | Cte d'Ivoire          | Wine           | 1.62          |
| 3 | 1986 | Americas        | Colombia              | Beer           | 4.27          |
| 4 | 1987 | Americas        | Saint Kitts and Nevis | Beer           | 1.98          |

```

World alcohol consumption sample data:
  Year  H0 region  Country Beverage Types  Display Value
0  1986  Western Pacific  Viet Nam      Wine      0.00
1  1986    Americas      Uruguay      Other      0.50
2  1985    Africa      Cte d'Ivoire    Wine      1.62
3  1986    Americas      Colombia      Beer      4.27
4  1987    Americas  Saint Kitts and Nevi  Beer      1.98

Shape of the dataframe: (5, 5)

Number of rows: 5

Number of column: 5

Extract Column Names:
Index(['Year', 'H0 region', 'Country', 'Beverage Types', 'Display Value'], dtype='object')

```

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



```
Original DataFrame:
name_code date_of_birth age
0 c0001 12/05/2002 18.5
1 1000c 16/02/1999 21.2
2 b00c2 25/09/1998 22.5
3 b2c02 12/02/2022 22.0
4 c2222 15/09/1997 23.0

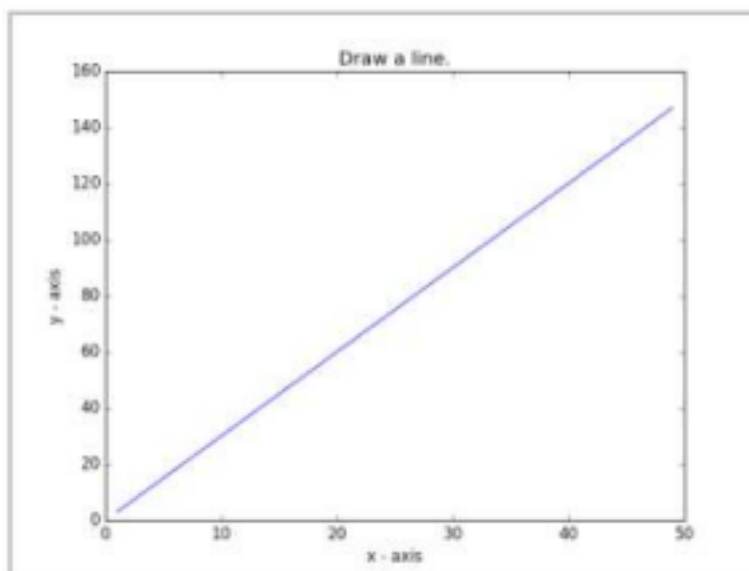
Index of a substring in a specified column of a dataframe:
name_code date_of_birth age Index
0 c0001 12/05/2002 18.5 0
1 1000c 16/02/1999 21.2 4
2 b00c2 25/09/1998 22.5 3
3 b2c02 12/02/2022 22.0 2
4 c2222 15/09/1997 23.0 0
```

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

```
Original DataFrame:
company_code date_of_sale sale_amount
0 Abcd 12/05/2002 12348.5
1 EFGF 16/02/1999 233331.2
2 zefsalf 25/09/1998 22.5
3 sdfslw 12/02/2022 2566552.0
4 zekfsdf 15/09/1997 23.0

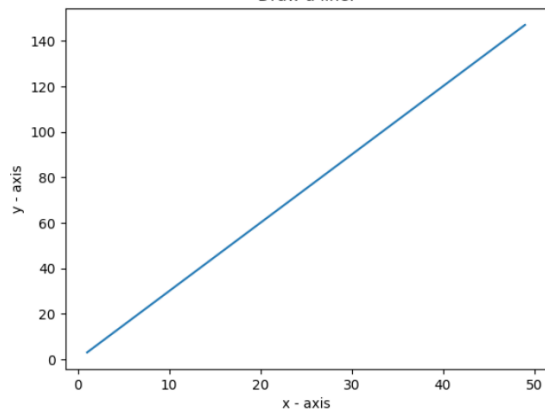
Swap cases in company_code:
company_code date_of_sale sale_amount swapped_company_code
0 Abcd 12/05/2002 12348.5 aBCD
1 EFGF 16/02/1999 233331.2 efgf
2 zefsalf 25/09/1998 22.5 ZEFSALF
3 sdfslw 12/02/2022 2566552.0 SDFSLEW
4 zekfsdf 15/09/1997 23.0 ZEKFSDF
```

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





↩ Values of X:  
 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49  
 Values of Y (thrice of X):  
 [3, 6, 9, 12, 15, 18, 21, 24, 27, 30, 33, 36, 39, 42, 45, 48, 51, 54, 57, 60, 63, 66, 69, 72, 75, 78, 81, 84, 87, 90, 93, 96, 99, 102, 105, 108, 111, 114, 117, 120, 123, 126, 129, 132, 135, 138, 141, 144, 147, 150, 153, 156, 159, 162, 165, 168, 171, 174, 177, 180, 183, 186, 189, 192, 195, 198, 201, 204, 207, 210, 213, 216, 219, 222, 225, 228, 231, 234, 237, 240, 243, 246, 249, 252, 255, 258, 261, 264, 267, 270, 273, 276, 279, 282, 285, 288, 291, 294, 297, 300, 303, 306, 309, 312, 315, 318, 321, 324, 327, 330, 333, 336, 339, 342, 345, 348, 351, 354, 357, 360, 363, 366, 369, 372, 375, 378, 381, 384, 387, 390, 393, 396, 399, 402, 405, 408, 411, 414, 417, 420, 423, 426, 429, 432, 435, 438, 441, 444, 447, 450, 453, 456, 459, 462, 465, 468, 471, 474, 477, 480, 483, 486, 489, 492, 495, 498, 501, 504, 507, 510, 513, 516, 519, 522, 525, 528, 531, 534, 537, 540, 543, 546, 549, 552, 555, 558, 561, 564, 567, 570, 573, 576, 579, 582, 585, 588, 591, 594, 597, 600, 603, 606, 609, 612, 615, 618, 621, 624, 627, 630, 633, 636, 639, 642, 645, 648, 651, 654, 657, 660, 663, 666, 669, 672, 675, 678, 681, 684, 687, 690, 693, 696, 699, 702, 705, 708, 711, 714, 717, 720, 723, 726, 729, 732, 735, 738, 741, 744, 747, 750, 753, 756, 759, 762, 765, 768, 771, 774, 777, 780, 783, 786, 789, 792, 795, 798, 801, 804, 807, 810, 813, 816, 819, 822, 825, 828, 831, 834, 837, 840, 843, 846, 849, 852, 855, 858, 861, 864, 867, 870, 873, 876, 879, 882, 885, 888, 891, 894, 897, 900, 903, 906, 909, 912, 915, 918, 921, 924, 927, 930, 933, 936, 939, 942, 945, 948, 951, 954, 957, 960, 963, 966, 969, 972, 975, 978, 981, 984, 987, 990, 993, 996, 999, 1002, 1005, 1008, 1011, 1014, 1017, 1020, 1023, 1026, 1029, 1032, 1035, 1038, 1041, 1044, 1047, 1050, 1053, 1056, 1059, 1062, 1065, 1068, 1071, 1074, 1077, 1080, 1083, 1086, 1089, 1092, 1095, 1098, 1101, 1104, 1107, 1110, 1113, 1116, 1119, 1122, 1125, 1128, 1131, 1134, 1137, 1140, 1143, 1146, 1149, 1152, 1155, 1158, 1161, 1164, 1167, 1170, 1173, 1176, 1179, 1182, 1185, 1188, 1191, 1194, 1197, 1200, 1203, 1206, 1209, 1212, 1215, 1218, 1221, 1224, 1227, 1230, 1233, 1236, 1239, 1242, 1245, 1248, 1251, 1254, 1257, 1260, 1263, 1266, 1269, 1272, 1275, 1278, 1281, 1284, 1287, 1290, 1293, 1296, 1299, 1302, 1305, 1308, 1311, 1314, 1317, 1320, 1323, 1326, 1329, 1332, 1335, 1338, 1341, 1344, 1347, 1350, 1353, 1356, 1359, 1362, 1365, 1368, 1371, 1374, 1377, 1380, 1383, 1386, 1389, 1392, 1395, 1398, 1401, 1404, 1407, 1410, 1413, 1416, 1419, 1422, 1425, 1428, 1431, 1434, 1437, 1440, 1443, 1446, 1449, 1452, 1455, 1458, 1461, 1464, 1467, 1470, 1473, 1476, 1479, 1482, 1485, 1488, 1491, 1494, 1497, 1500, 1503, 1506, 1509, 1512, 1515, 1518, 1521, 1524, 1527, 1530, 1533, 1536, 1539, 1542, 1545, 1548, 1551, 1554, 1557, 1560, 1563, 1566, 1569, 1572, 1575, 1578, 1581, 1584, 1587, 1590, 1593, 1596, 1599, 1602, 1605, 1608, 1611, 1614, 1617, 1620, 1623, 1626, 1629, 1632, 1635, 1638, 1641, 1644, 1647, 1650, 1653, 1656, 1659, 1662, 1665, 1668, 1671, 1674, 1677, 1680, 1683, 1686, 1689, 1692, 1695, 1698, 1701, 1704, 1707, 1710, 1713, 1716, 1719, 1722, 1725, 1728, 1731, 1734, 1737, 1740, 1743, 1746, 1749, 1752, 1755, 1758, 1761, 1764, 1767, 1770, 1773, 1776, 1779, 1782, 1785, 1788, 1791, 1794, 1797, 1800, 1803, 1806, 1809, 1812, 1815, 1818, 1821, 1824, 1827, 1830, 1833, 1836, 1839, 1842, 1845, 1848, 1851, 1854, 1857, 1860, 1863, 1866, 1869, 1872, 1875, 1878, 1881, 1884, 1887, 1890, 1893, 1896, 1899, 1902, 1905, 1908, 1911, 1914, 1917, 1920, 1923, 1926, 1929, 1932, 1935, 1938, 1941, 1944, 1947, 1950, 1953, 1956, 1959, 1962, 1965, 1968, 1971, 1974, 1977, 1980, 1983, 1986, 1989, 1992, 1995, 1998, 2001, 2004, 2007, 2010, 2013, 2016, 2019, 2022, 2025, 2028, 2031, 2034, 2037, 2040, 2043, 2046, 2049, 2052, 2055, 2058, 2061, 2064, 2067, 2070, 2073, 2076, 2079, 2082, 2085, 2088, 2091, 2094, 2097, 2100, 2103, 2106, 2109, 2112, 2115, 2118, 2121, 2124, 2127, 2130, 2133, 2136, 2139, 2142, 2145, 2148, 2151, 2154, 2157, 2160, 2163, 2166, 2169, 2172, 2175, 2178, 2181, 2184, 2187, 2190, 2193, 2196, 2199, 2202, 2205, 2208, 2211, 2214, 2217, 2220, 2223, 2226, 2229, 2232, 2235, 2238, 2241, 2244, 2247, 2250, 2253, 2256, 2259, 2262, 2265, 2268, 2271, 2274, 2277, 2280, 2283, 2286, 2289, 2292, 2295, 2298, 2301, 2304, 2307, 2310, 2313, 2316, 2319, 2322, 2325, 2328, 2331, 2334, 2337, 2340, 2343, 2346, 2349, 2352, 2355, 2358, 2361, 2364, 2367, 2370, 2373, 2376, 2379, 2382, 2385, 2388, 2391, 2394, 2397, 2400, 2403, 2406, 2409, 2412, 2415, 2418, 2421, 2424, 2427, 2430, 2433, 2436, 2439, 2442, 2445, 2448, 2451, 2454, 2457, 2460, 2463, 2466, 2469, 2472, 2475, 2478, 2481, 2484, 2487, 2490, 2493, 2496, 2499, 2502, 2505, 2508, 2511, 2514, 2517, 2520, 2523, 2526, 2529, 2532, 2535, 2538, 2541, 2544, 2547, 2550, 2553, 2556, 2559, 2562, 2565, 2568, 2571, 2574, 2577, 2580, 2583, 2586, 2589, 2592, 2595, 2598, 2601, 2604, 2607, 2610, 2613, 2616, 2619, 2622, 2625, 2628, 2631, 2634, 2637, 2640, 2643, 2646, 2649, 2652, 2655, 2658, 2661, 2664, 2667, 2670, 2673, 2676, 2679, 2682, 2685, 2688, 2691, 2694, 2697, 2700, 2703, 2706, 2709, 2712, 2715, 2718, 2721, 2724, 2727, 2730, 2733, 2736, 2739, 2742, 2745, 2748, 2751, 2754, 2757, 2760, 2763, 2766, 2769, 2772, 2775, 2778, 2781, 2784, 2787, 2790, 2793, 2796, 2799, 2802, 2805, 2808, 2811, 2814, 2817, 2820, 2823, 2826, 2829, 2832, 2835, 2838, 2841, 2844, 2847, 2850, 2853, 2856, 2859, 2862, 2865, 2868, 2871, 2874, 2877, 2880, 2883, 2886, 2889, 2892, 2895, 2898, 2901, 2904, 2907, 2910, 2913, 2916, 2919, 2922, 2925, 2928, 2931, 2934, 2937, 2940, 2943, 2946, 2949, 2952, 2955, 2958, 2961, 2964, 2967, 2970, 2973, 2976, 2979, 2982, 2985, 2988, 2991, 2994, 2997, 3000, 3003, 3006, 3009, 3012, 3015, 3018, 3021, 3024, 3027, 3030, 3033, 3036, 3039, 3042, 3045, 3048, 3051, 3054, 3057, 3060, 3063, 3066, 3069, 3072, 3075, 3078, 3081, 3084, 3087, 3090, 3093, 3096, 3099, 3102, 3105, 3108, 3111, 3114, 3117, 3120, 3123, 3126, 3129, 3132, 3135, 3138, 3141, 3144, 3147, 3150, 3153, 3156, 3159, 3162, 3165, 3168, 3171, 3174, 3177, 3180, 3183, 3186, 3189, 3192, 3195, 3198, 3201, 3204, 3207, 3210, 3213, 3216, 3219, 3222, 3225, 3228, 3231, 3234, 3237, 3240, 3243, 3246, 3249, 3252, 3255, 3258, 3261, 3264, 3267, 3270, 3273, 3276, 3279, 3282, 3285, 3288, 3291, 3294, 3297, 3300, 3303, 3306, 3309, 3312, 3315, 3318, 3321, 3324, 3327, 3330, 3333, 3336, 3339, 3342, 3345, 3348, 3351, 3354, 3357, 3360, 3363, 3366, 3369, 3372, 3375, 3378, 3381, 3384, 3387, 3390, 3393, 3396, 3399, 3402, 3405, 3408, 3411, 3414, 3417, 3420, 3423, 3426, 3429, 3432, 3435, 3438, 3441, 3444, 3447, 3450, 3453, 3456, 3459, 3462, 3465, 3468, 3471, 3474, 3477, 3480, 3483, 3486, 3489, 3492, 3495, 3498, 3501, 3504, 3507, 3510, 3513, 3516, 3519, 3522, 3525, 3528, 3531, 3534, 3537, 3540, 3543, 3546, 3549, 3552, 3555, 3558, 3561, 3564, 3567, 3570, 3573, 3576, 3579, 3582, 3585, 3588, 3591, 3594, 3597, 3600, 3603, 3606, 3609, 3612, 3615, 3618, 3621, 3624, 3627, 3630, 3633, 3636, 3639, 3642, 3645, 3648, 3651, 3654, 3657, 3660, 3663, 3666, 3669, 3672, 3675, 3678, 3681, 3684, 3687, 3690, 3693, 3696, 3699, 3702, 3705, 3708, 3711, 3714, 3717, 3720, 3723, 3726, 3729, 3732, 3735, 3738, 3741, 3744, 3747, 3750, 3753, 3756, 3759, 3762, 3765, 3768, 3771, 3774, 3777, 3780, 3783, 3786, 3789, 3792, 3795, 3798, 3801, 3804, 3807, 3810, 3813, 3816, 3819, 3822, 3825, 3828, 3831, 3834, 3837, 3840, 3843, 3846, 3849, 3852, 3855, 3858, 3861, 3864, 3867, 3870, 3873, 3876, 3879, 3882, 3885, 3888, 3891, 3894, 3897, 3900, 3903, 3906, 3909, 3912, 3915, 3918, 3921, 3924, 3927, 3930, 3933, 3936, 3939, 3942, 3945, 3948, 3951, 3954, 3957, 3960, 3963, 3966, 3969, 3972, 3975, 3978, 3981, 3984, 3987, 3990, 3993, 3996, 4000]



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.

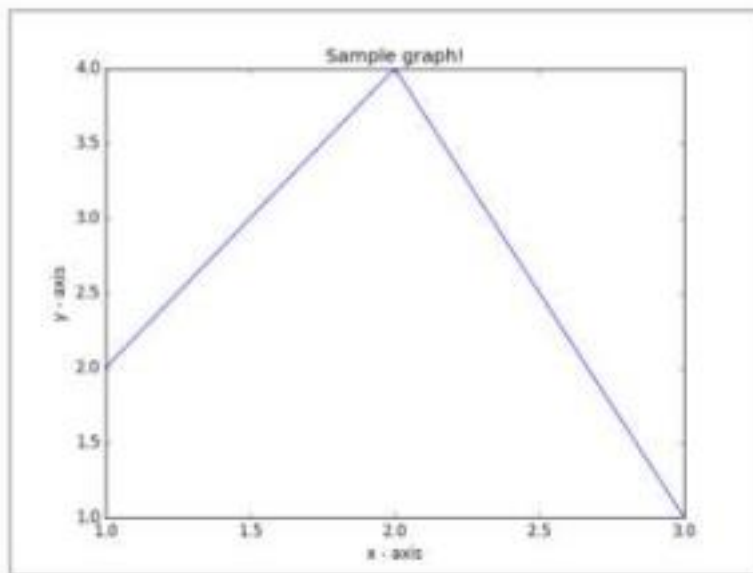
*Test Data:*

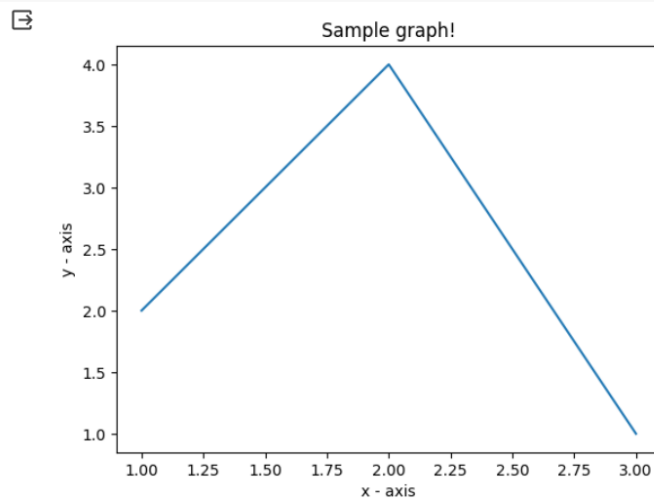
test.txt

1 2

2 4

3 1





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

Sample Financial data (fdata.csv):

Date,Open,High,Low,Close

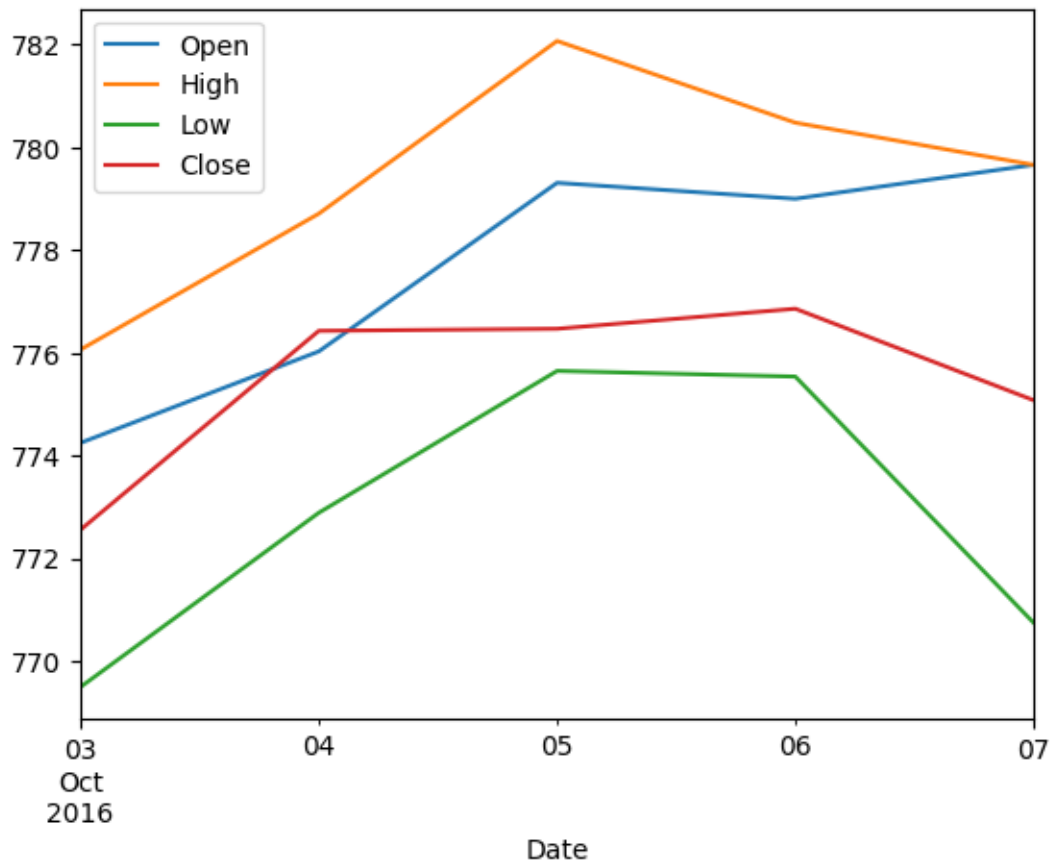
10-03-16,774.25,776.065002,769.5,772.559998

10-04-16,776.030029,778.710022,772.890015,776.429993

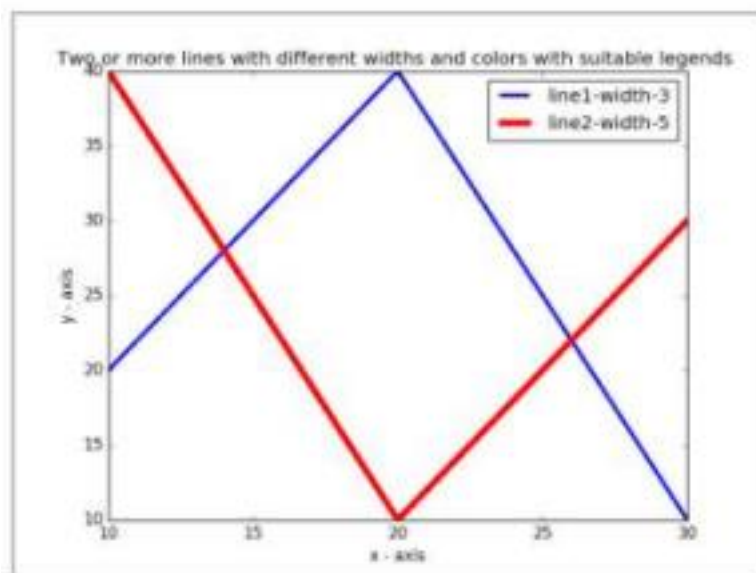
10-05-16,779.309998,782.070007,775.650024,776.469971

10-06-16,779,780.47998,775.539978,776.859985

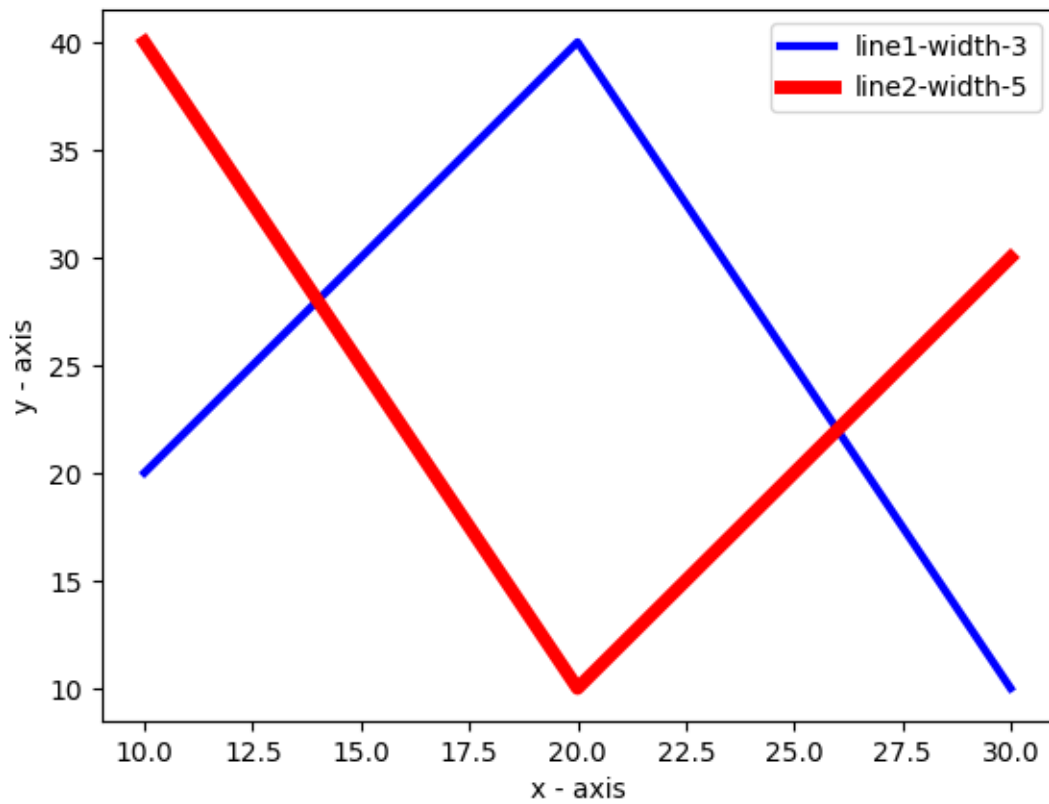
10-07-16,779.659973,779.659973,770.75,775.080017



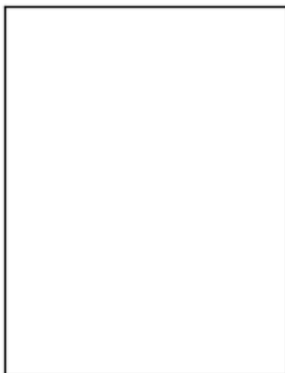
25. Write a Python program to plot two or more lines with legends, different widths and colors.



Two or more lines with different widths and colors with suitable legends



26. Write a Python program to create multiple plots.



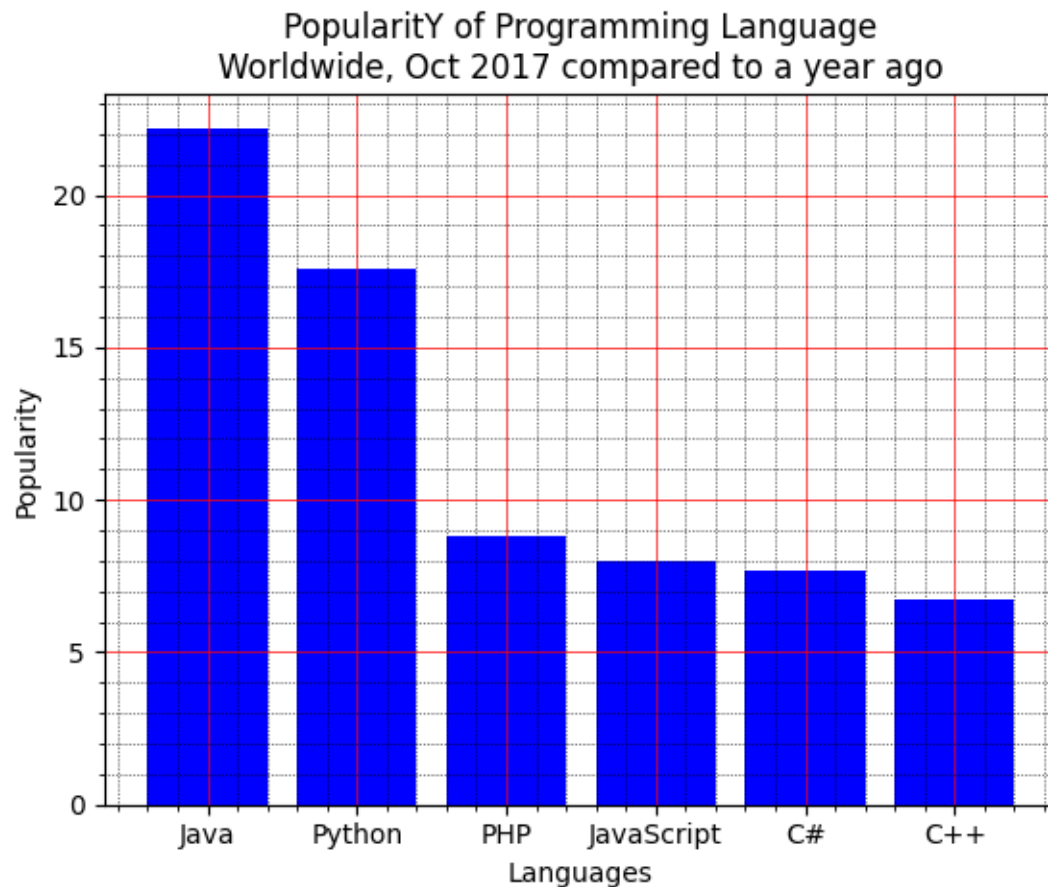
27. Write a Python programming to display a bar chart of the popularity of

programming Languages.

Sample data:

Programming languages: Java, Python, PHP, JavaScript, C#, C++

Popularity: 22.2, 17.6, 8.8, 8, 7.7, 6.7

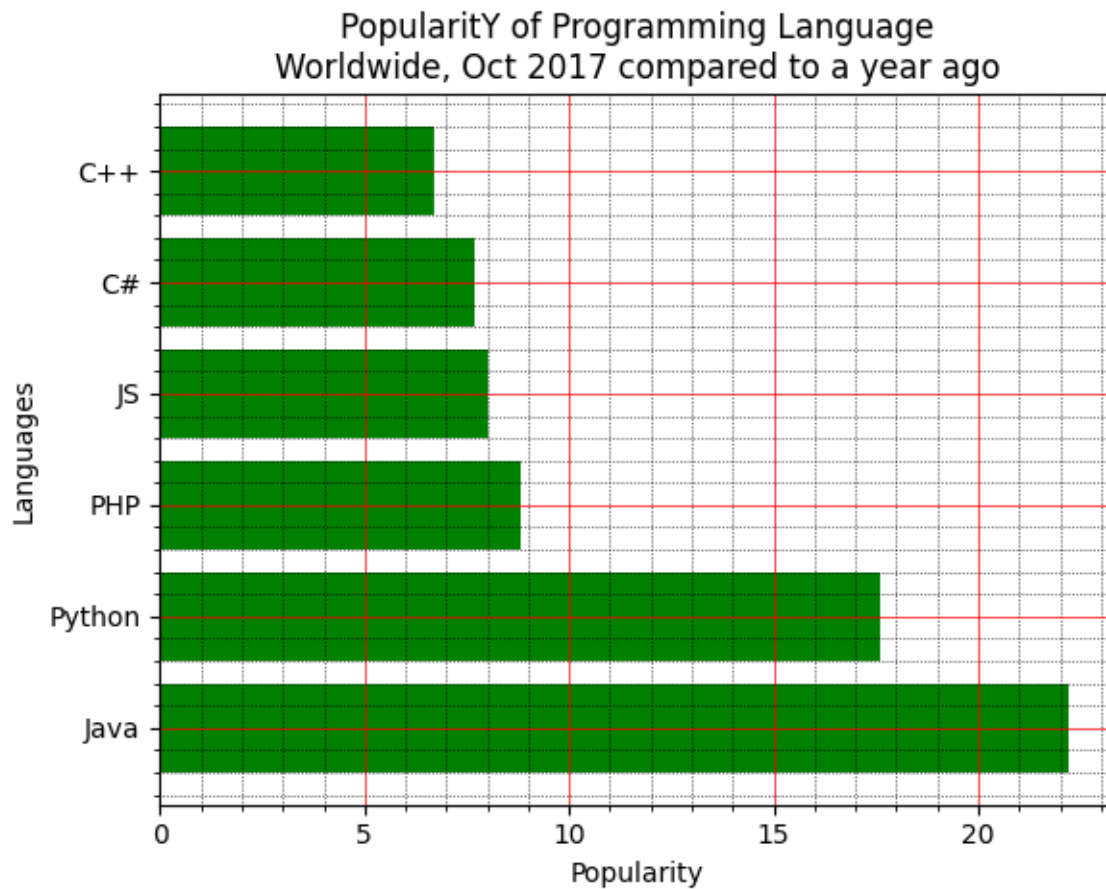


28. Write a Python programming to display a horizontal bar chart of the popularity of programming Languages.

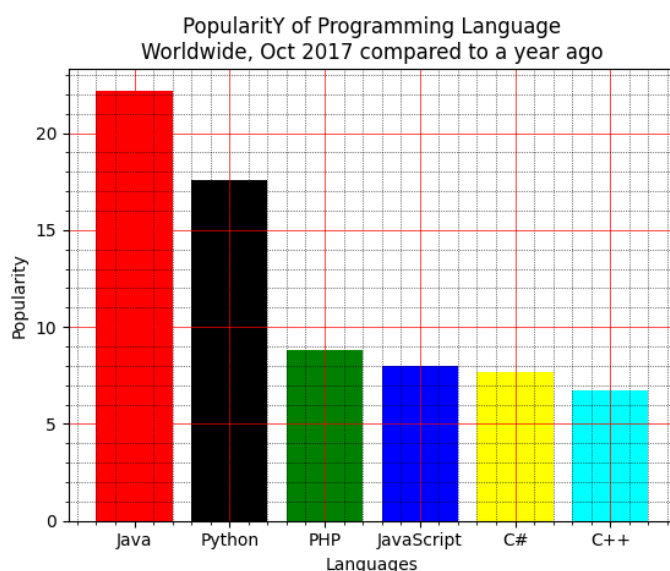
Sample data:

Programming languages: Java, Python, PHP, JavaScript, C#, C++

Popularity: 22.2, 17.6, 8.8, 8, 7.7, 6.7



29. Write a Python program to display a bar chart of the popularity of programming Languages. Use different color for each bar. Sample data:  
 Programming languages: Java, Python, PHP, JavaScript, C#, C++  
 Popularity: 22.2, 17.6, 8.8, 8, 7.7, 6.7



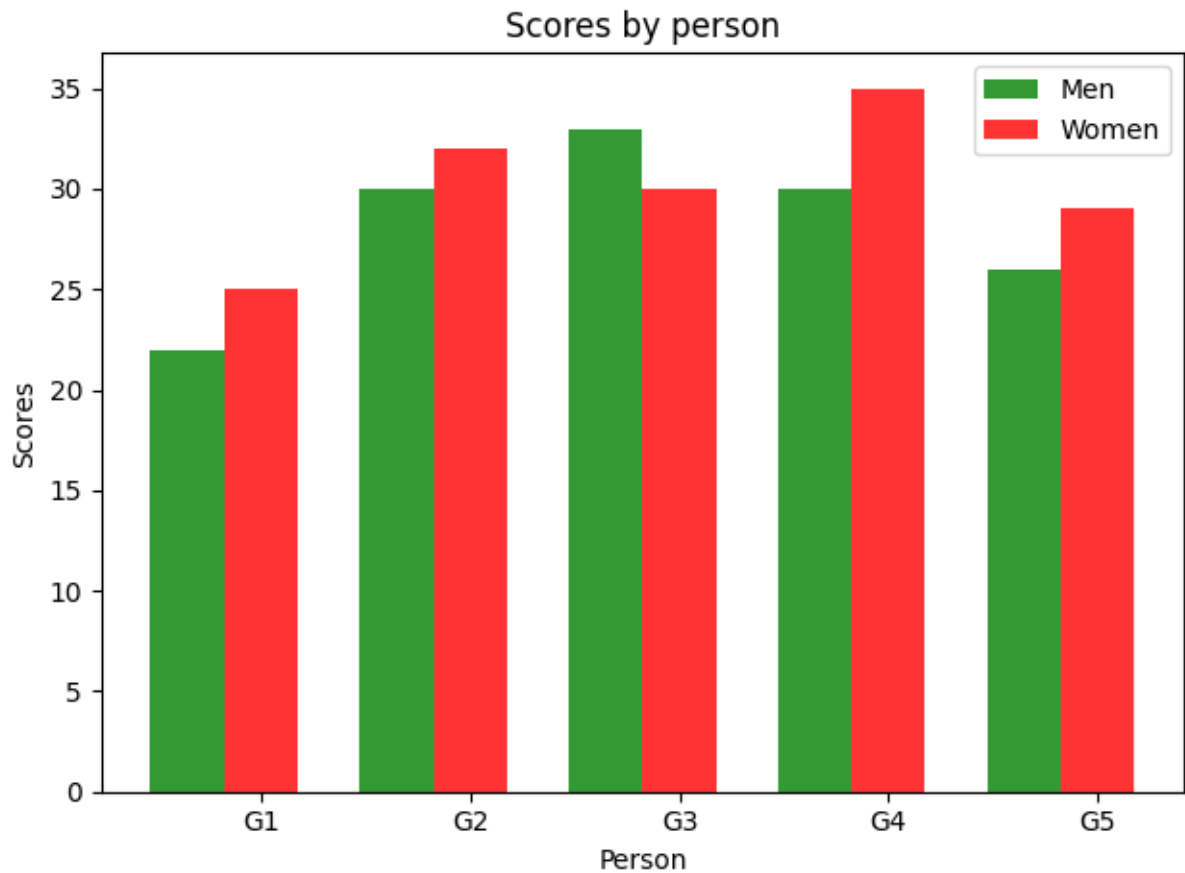
30. Write a Python program to create bar plot of scores by group and gender.

Use multiple X values on the same chart for men and women.

Sample Data:

Means (men) = (22, 30, 35, 35, 26)

Means (women) = (25, 32, 30, 35, 29)



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

Note: Use bottom to stack the women's bars on top of the men's bars.

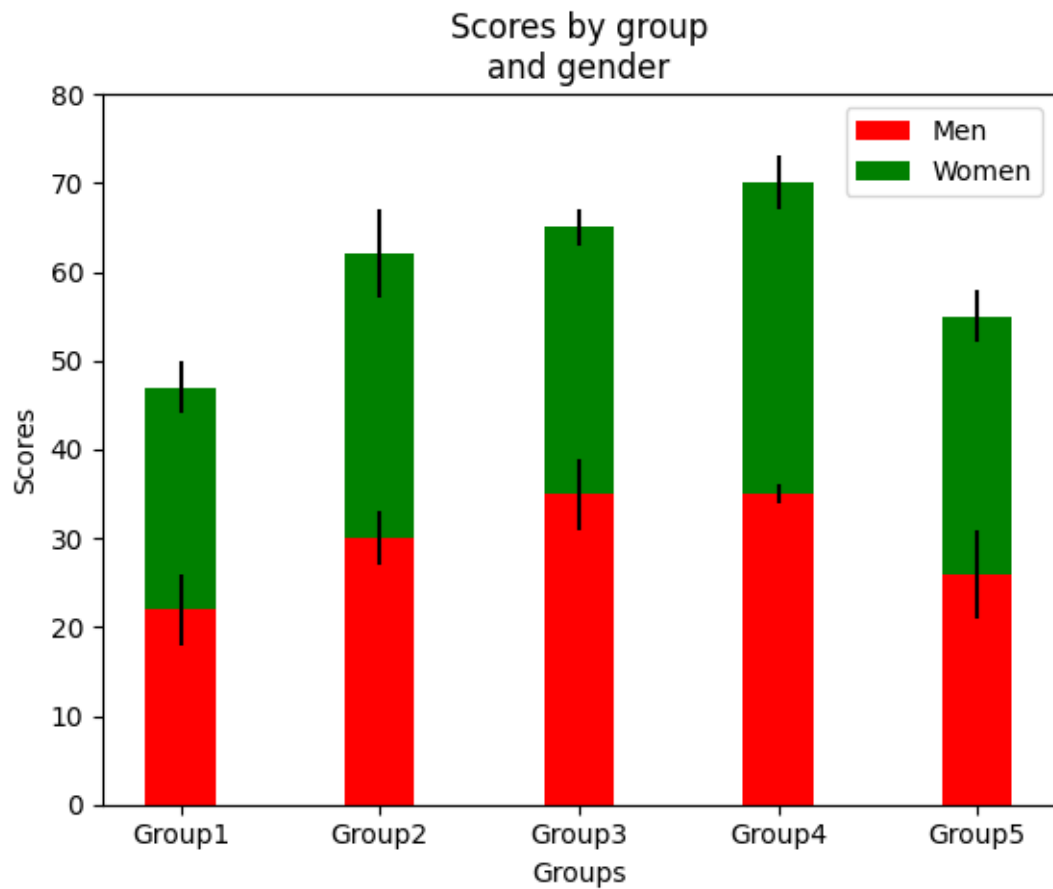
Sample Data:

Means (men) = (22, 30, 35, 35, 26)

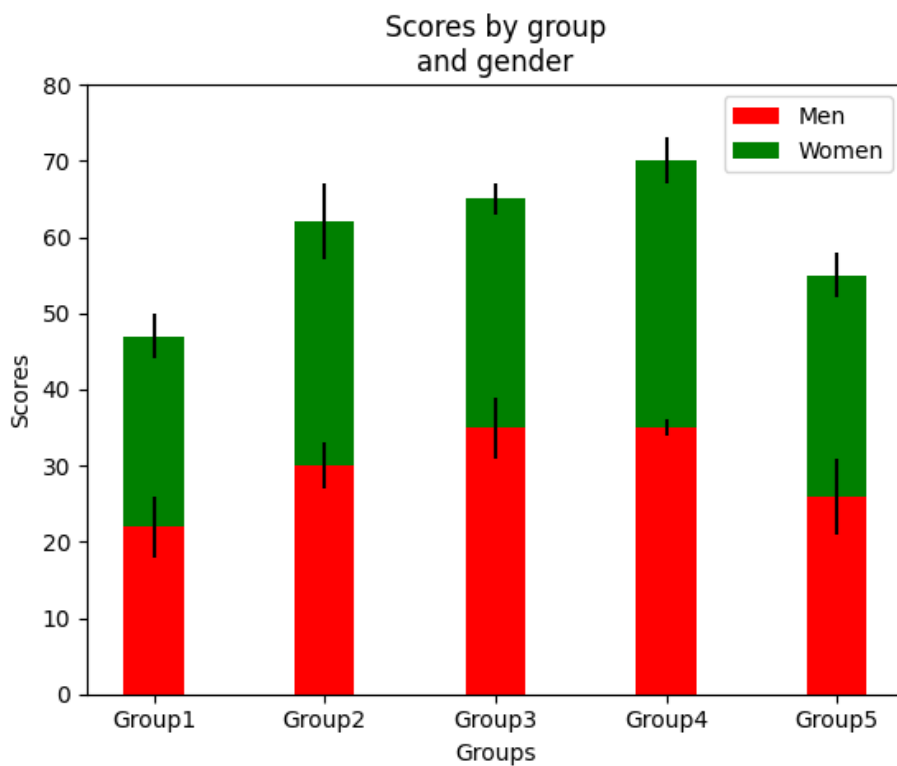
Means (women) = (25, 32, 30, 35, 29)

Men Standard deviation = (4, 3, 4, 1, 5)

Women Standard deviation = (3, 5, 2, 3, 3)

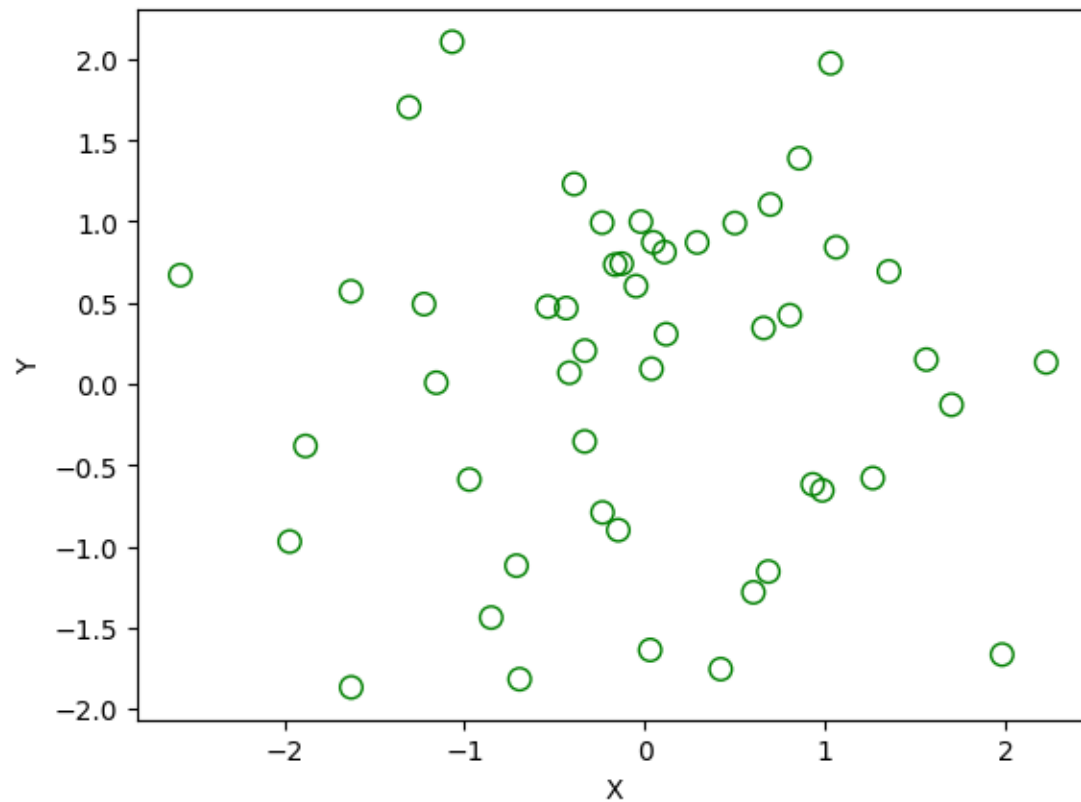


32. Write a Python program to draw a scatter graph taking a random distribution in X and Y and plotted against each other.

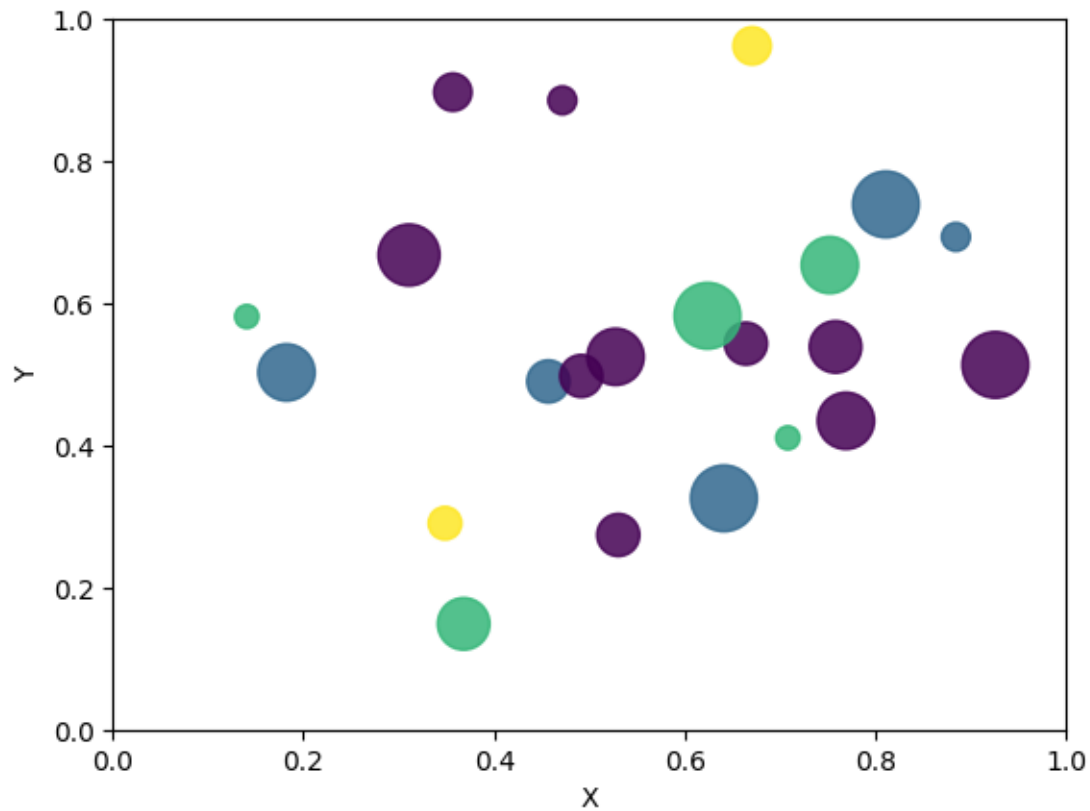




33. Write a Python program to draw a scatter plot with empty circles taking a random distribution in X and Y and plotted against each other.



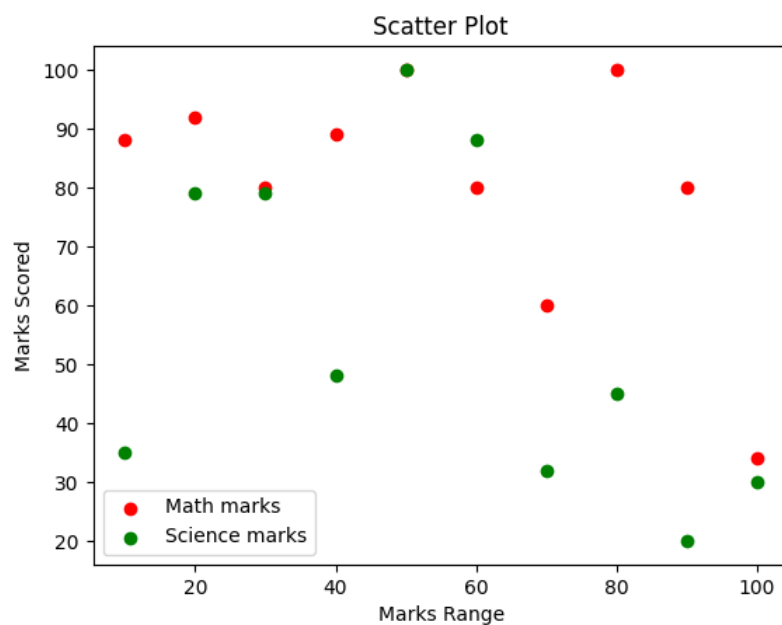
34. Write a Python program to draw a scatter plot using random distributions to generate balls of different sizes.



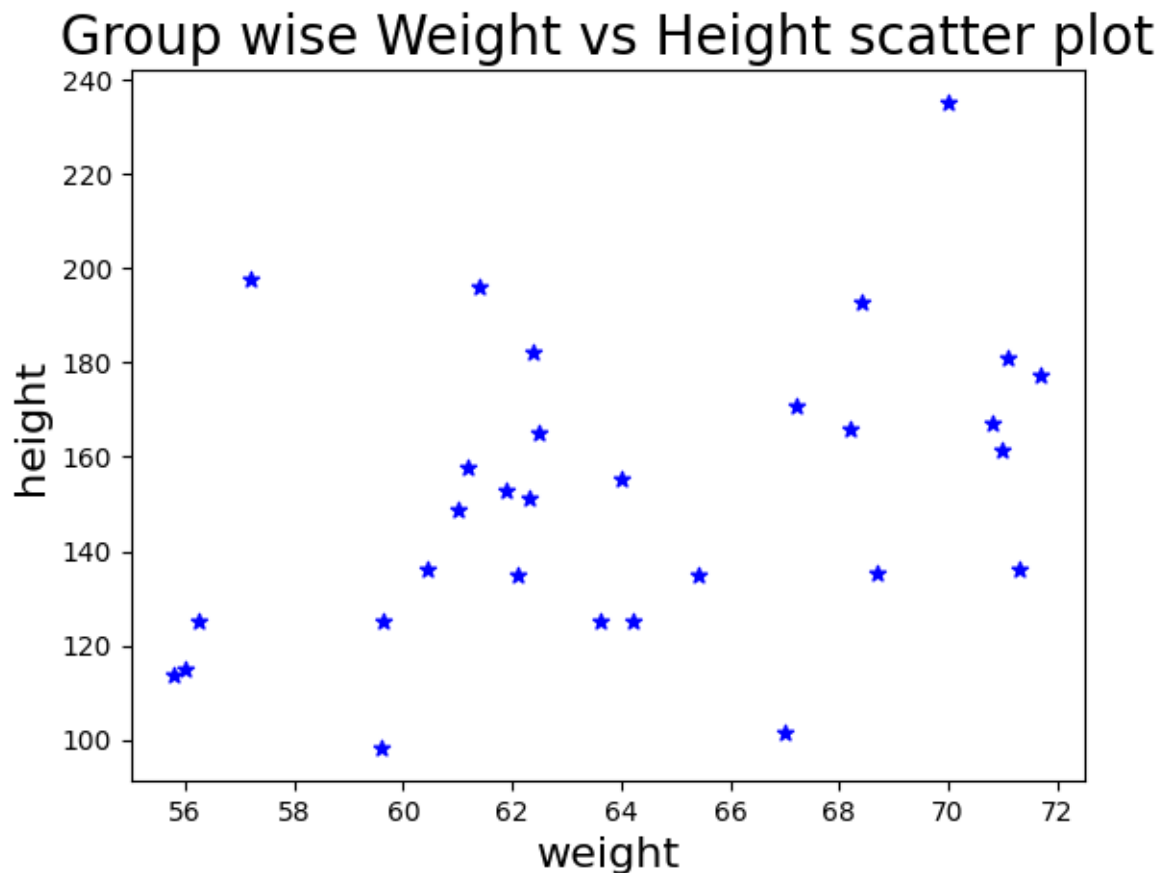
35. Write a Python program to draw a scatter plot comparing two subject marks of Mathematics and Science. Use marks of 10 students. Sample data:

Test Data:

```
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]
```



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



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

Sample data: {'X':[78,85,96,80,86], 'Y':[84,94,89,83,86], 'Z':[86,97,96,72,83]}

```
[50]: #37. Write a Pandas program to create a dataframe from a dictionary and display it.
#Sample data: {'X':[78,85,96,80,86], 'Y':[84,94,89,83,86], 'Z':[86,97,96,72,83]}
import pandas as pd
df = pd.DataFrame({'X':[78,85,96,80,86], 'Y':[84,94,89,83,86], 'Z':[86,97,96,72,83]});
print(df)
```

|   | 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.

Sample Python dictionary data and list labels:

```
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)
```

|   | 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.

Sample Python dictionary data and list labels:

```
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']
```

|   | 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.

Sample Python dictionary data and list labels:

```
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']
```



Select specific columns:

|   | 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  |