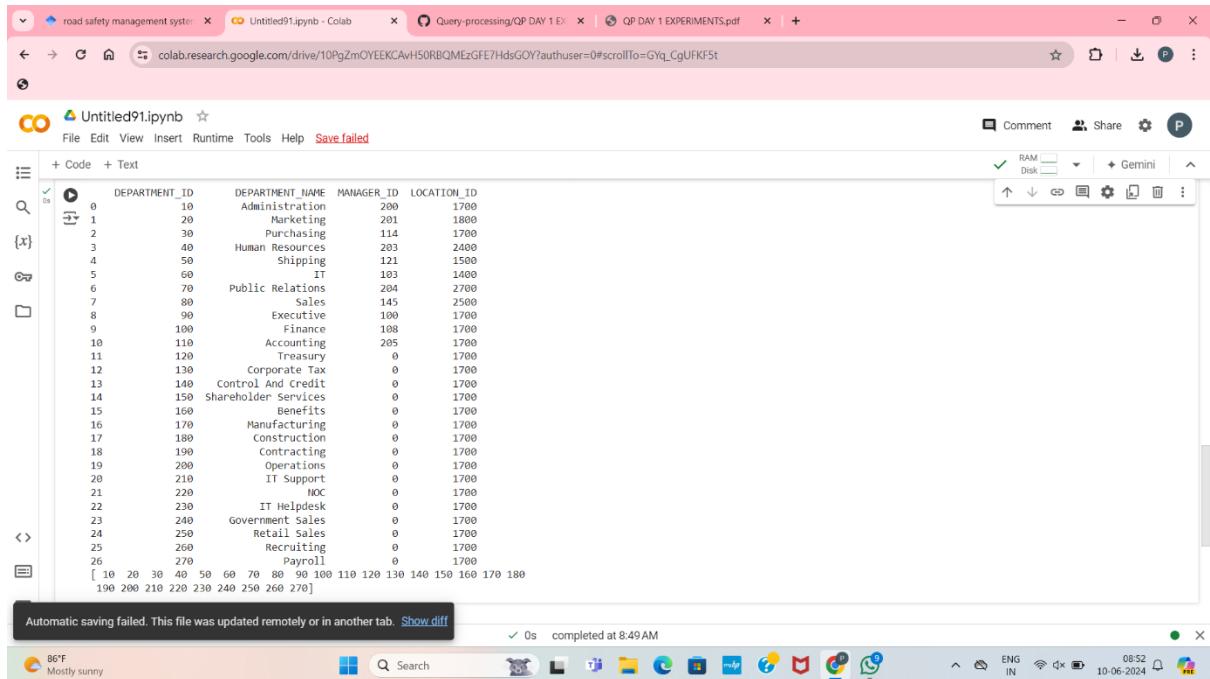


1. Write a Pandas program to select distinct department id from employees file.

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
10	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

CODE:

OUTPUT:



The screenshot shows a Google Colab interface with multiple tabs at the top: 'road safety management system', 'Untitled91.ipynb - Colab', 'Query-processing/QP DAY 1 E...', and 'QP DAY 1 EXPERIMENTS.pdf'. The main area displays a Pandas DataFrame titled 'Untitled91.ipynb'. The DataFrame has columns: DEPARTMENT_ID, DEPARTMENT_NAME, MANAGER_ID, and LOCATION_ID. The data includes various departments like Administration, Marketing, Purchasing, Human Resources, Shipping, 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, and Payroll. The code cell above the table contains a list of department IDs from 10 to 270. A status bar at the bottom indicates 'Automatic saving failed. This file was updated remotely or in another tab. Show diff?' and 'completed at 8:49 AM'.

	DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
0	10	Administration	200	1700
1	20	Marketing	201	1800
2	30	Purchasing	114	1700
3	40	Human Resources	203	2400
4	50	Shipping	121	1500
5	60	IT	103	1400
6	70	Public Relations	204	2700
7	80	Sales	145	2500
8	90	Executive	100	1700
9	100	Finance	108	1700
10	110	Accounting	205	1700
11	120	Treasury	0	1700
12	130	Corporate Tax	0	1700
13	140	Control And Credit	0	1700
14	150	Shareholder Services	0	1700
15	160	Benefits	0	1700
16	170	Manufacturing	0	1700
17	180	Construction	0	1700
18	190	Contracting	0	1700
19	200	Operations	0	1700
20	210	IT Support	0	1700
21	220	NOC	0	1700
22	230	IT Helpdesk	0	1700
23	240	Government Sales	0	1700
24	250	Retail Sales	0	1700
25	260	Recruiting	0	1700
26	270	Payroll	0	1700

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 |
+-----+-----+-----+-----+-----+
| 102 | 2001-01-13 | 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 |
+-----+-----+-----+-----+-----+
```

```

Untitled91.ipynb  ☆
File Edit View Insert Runtime Tools Help Save failed
+ Code + Text
[3] 26      270      Payroll      0      1700
 [ 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 ]
{x}
import pandas as pd
# Sample data
data = [
    'EMPLOYEE_ID': [102, 101, 101, 201, 114, 122, 200, 176, 176, 200],
    'START_DATE': ['2001-01-13', '1997-09-21', '2004-02-28', '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', '2001-06-17', '2006-12-31', '2007-12-31', '2006-12-31'],
    'JOB_ID': ['IT_PROG', 'AC_ACCOUNT', 'AC_MGR', 'MK_REP', 'ST_CLERK', 'AD_ASST', 'SA_REP', 'SA_MAN', 'AC_ACCOUNT'],
    'DEPARTMENT_ID': [60, 110, 110, 20, 50, 50, 80, 80, 90]
]
# Create DataFrame
df = pd.DataFrame(data)

# Count the number of jobs per employee
job_counts = df['EMPLOYEE_ID'].value_counts()

# Filter employees with two or more jobs
employees_with_multiple_jobs = job_counts[job_counts >= 2].index

# Convert to DataFrame for better presentation
employees_with_multiple_jobs_df = pd.DataFrame(employees_with_multiple_jobs, columns=['EMPLOYEE_ID'])

# Display the result
print(employees_with_multiple_jobs_df)

```

0s completed at 9:02 AM

86°F Mostly sunny

OUTPUT:

```

Untitled91.ipynb  ☆
File Edit View Insert Runtime Tools Help Save failed
+ Code + Text
job_counts = df['EMPLOYEE_ID'].value_counts()
# Filter employees with two or more jobs
employees_with_multiple_jobs = job_counts[job_counts >= 2].index
# Convert to DataFrame for better presentation
employees_with_multiple_jobs_df = pd.DataFrame(employees_with_multiple_jobs, columns=['EMPLOYEE_ID'])

# Display the result
print(employees_with_multiple_jobs_df)

```

EMPLOYEE_ID
0 101
1 200
2 176

0s completed at 9:02 AM

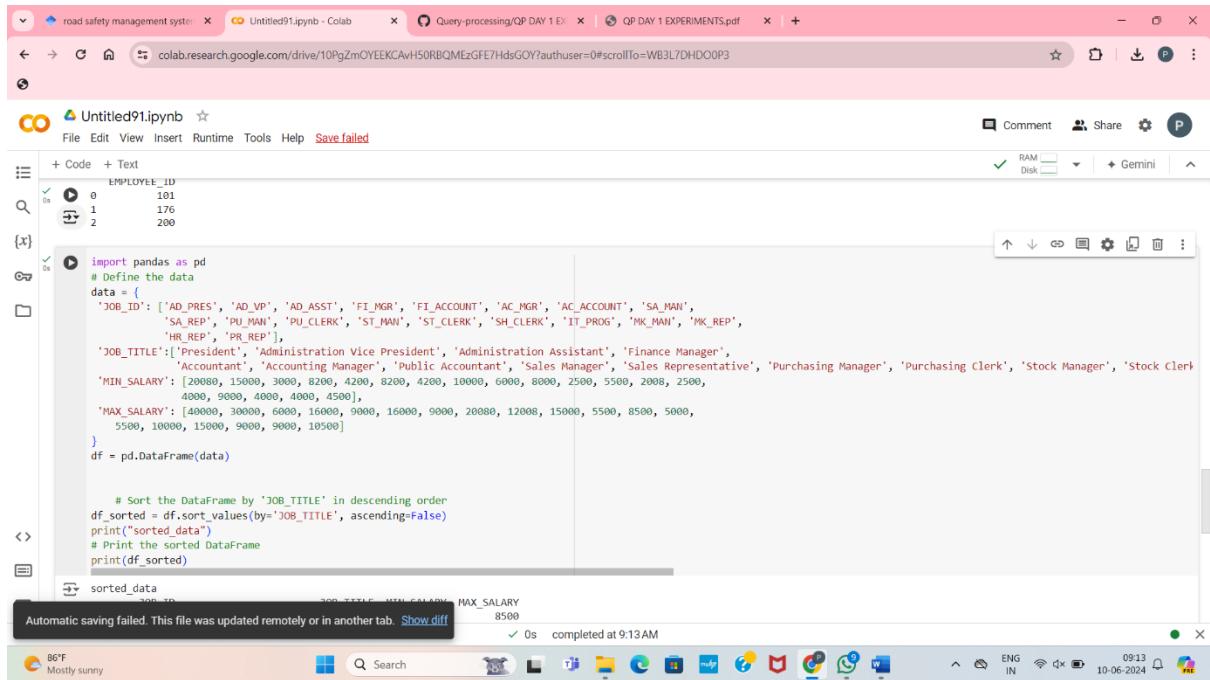
86°F Mostly sunny

1. 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

CODE:



```

Untitled91.ipynb  ☆
File Edit View Insert Runtime Tools Help Save failed
+ Code + Text
EMPLOYEE_ID
0 101
1 176
2 200
{x}
import pandas as pd
# Define the data
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',
                  '4000, 9000, 4000, 4000, 4500],
    'MIN_SALARY': [20080, 15000, 3000, 8200, 4200, 10000, 6000, 8000, 2500, 2800, 2500,
                  4000, 9000, 4000, 4000, 4500],
    'MAX_SALARY': [40000, 30000, 6000, 16000, 9000, 20080, 12000, 15000, 5500, 8500, 5000,
                  5500, 10000, 15000, 9000, 9000, 10500]
}
df = pd.DataFrame(data)

# Sort the DataFrame by 'JOB_TITLE' in descending order
df_sorted = df.sort_values(by='JOB_TITLE', ascending=False)
print("sorted_data")
# Print the sorted DataFrame
print(df_sorted)

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

```

Automatic saving failed. This file was updated remotely or in another tab. Show diff

The screenshot shows a Jupyter Notebook interface in Google Colab. The code cell contains the following Python code:

```
print(df_sorted)
```

The output of the code is a sorted DataFrame:

	JOB_ID	JOB_TITLE	MIN_SALARY	MAX_SALARY
11	ST_MAN	Stock Manager	5500	8500
12	ST_CLERK	Stock Clerk	2000	5000
13	SH_CLERK	Shipping Clerk	2500	5500
8	SA REP	Sales Representative	6000	12000
7	SA MAN	Sales Manager	10000	20000
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	20000	40000
16	MKT REP	Marketing Representative	4000	9000
15	MKT 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

At the bottom of the notebook, there is a message: "Automatic saving failed. This file was updated remotely or in another tab. Show diff".

The system tray at the bottom right shows the date as 10-06-2024, time as 09:13, and weather as 86°F Mostly sunny.

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

The screenshot shows a Jupyter Notebook interface with two tabs open: "Untitled91.ipynb - Colab" and "Query-processing/QP DAY 1 EXPERIMENTS.pdf".

Data Frame:

```
[13]   3    FT_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
{x}     4    FI_ACCOUNT  Accountant           4200     9000
```

Code Cell:

```
import yfinance as yf
import pandas as pd
import matplotlib.pyplot as plt
# Define the ticker symbol for Alphabet Inc. (GOOGL)
ticker = 'GOOGL'
# Define the start and end dates
start_date = '2023-01-01'
end_date = '2023-10-01'
# Fetch historical data from Yahoo Finance
data = yf.download(ticker, start=start_date, end=end_date)
# Create a line plot
plt.figure(figsize=(10,6))
plt.plot(data['Close'], label='Close Price')
plt.title(f'Historical Stock Prices of {ticker} between {start_date} and {end_date}')
plt.xlabel('Date')
plt.ylabel('Price (USD)')
plt.legend()
plt.show()
```

Output:

100% [*****] 1 of 1 completed
Historical Stock Prices of GOOGL between 2023-01-01 and 2023-10-01

Automatic saving failed. This file was updated remotely or in another tab. Show diff

The notebook also displays a line plot titled "Historical Stock Prices of GOOGL between 2023-01-01 and 2023-10-01". The x-axis represents the date from January 2023 to October 2023, and the y-axis represents the price in USD, ranging from 90 to 140. The plot shows a general upward trend with significant volatility.

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

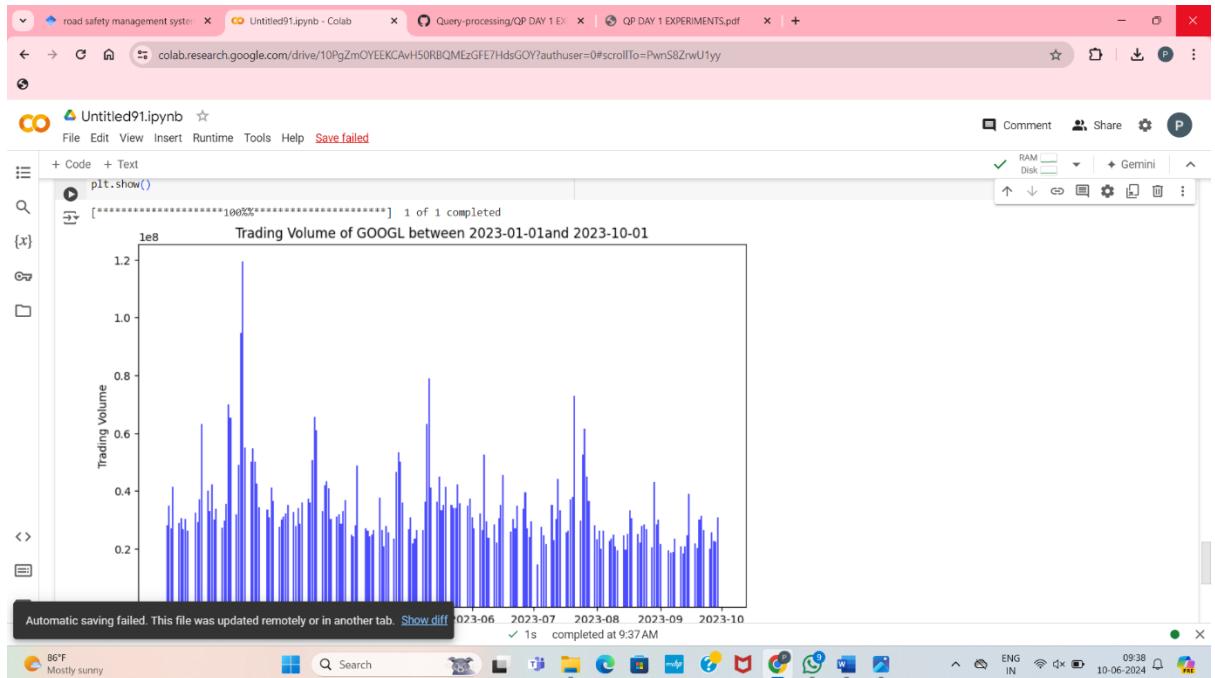
CODE:

The screenshot shows a Google Colab notebook titled "Untitled91.ipynb". The code cell contains the following Python script:

```
import pandas as pd
import yfinance as yf
import matplotlib.pyplot as plt
# Step 1: Retrieve historical stock data
ticker = "GOOGL"
# Ticker symbol for Alphabet Inc.
start_date = "2023-01-01"
end_date = "2023-10-01"
# Using yfinance to get the stock data
data = yf.download(ticker, start=start_date, end=end_date)
# Step 2: Filter the data for the desired date range
# Since we're interested in trading volume, we only need that
data = data["Volume"]
# Step 3: Create a bar plot
plt.figure(figsize=(10, 6))
plt.bar(data.index, data.values, color='blue', alpha=0.7)
plt.title(f"Trading Volume of {ticker} between {start_date} and {end_date}")
plt.xlabel("Date")
plt.ylabel("Trading Volume")
plt.show()
```

The output cell shows the command `plt.show()` and the resulting bar chart titled "Trading Volume of GOOGL between 2023-01-01 and 2023-10-01". The y-axis is labeled "Trading volume" and ranges from 0 to 1.2 with a multiplier of 10^8 . The x-axis shows dates from 2023-01-01 to 2023-10-01. The chart displays highly volatile trading volumes, with several sharp peaks exceeding 1.0.

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.

alphabet_stock_data:

CODE:

```

  road safety management system - Colab  Untitled91.ipynb - Colab  Query-processing/QP DAY 1 EX  |  QP DAY 1 EXPERIMENTS.pdf  |  what is the use of xtricks - Google Search
  colab.research.google.com/drive/10PgZmOYEEKCAvH50RBQMEzGFE7HdsGOY?authuser=0#scrollTo=5InuAYDXWNQV

  Untitled91.ipynb  ★
  File Edit View Insert Runtime Tools Help Save failed
  + Code + Text
  {x} 4s
  import pandas as pd
  import matplotlib.pyplot as plt
  # Creating a DataFrame from the provided data
  data = [
    'Date': ['01-04-2020', '02-04-2020', '03-04-2020', '06-04-2020', '07-04-2020', '08-04-2020', '09-04-2020',
    '13-04-2020', '14-04-2020', '15-04-2020', '16-04-2020', '17-04-2020', '20-04-2020', '21-04-2020',
    '22-04-2020', '23-04-2020', '24-04-2020', '27-04-2020', '28-04-2020', '29-04-2020', '30-04-2020',
    '01-05-2020'],
    'Open': [1122, 1098.26, 1119.015, 1138, 1221, 1206.5, 1224.08, 1209.18, 1245.09, 1245.61, 1274.1,
    1284.85, 1271, 1247, 1245.54, 1271.55, 1261.17, 1296, 1287.93, 1341.46, 1324.88, 1328.51],
    'High': [1129.69, 1126.86, 1123.54, 1194.66, 1225, 1219.07, 1225.57, 1220.51, 1282.07, 1280.46, 1279,
    1294.43, 1281.6, 1254.27, 1285.613, 1293.31, 1280.4, 1296.15, 1288.05, 1359.99, 1352.82, 1352.07],
    'Low': [1097.45, 1096.4, 1079.81, 1130.94, 1182.23, 1188.16, 1196.735, 1187.598, 1236.93, 1240.4,
    1242.62, 1271.23, 1261.37, 1209.71, 1242, 1265.67, 1249.45, 1269, 1232.2, 1325.34, 1322.49, 1311],
    'Close': [1105.62, 1120.84, 1097.88, 1186.92, 1186.51, 1210.28, 1211.45, 1217.56, 1269.23, 1262.47,
    1263.47, 1283.25, 1266.61, 1216.34, 1263.21, 1276.31, 1279.31, 1275.88, 1233.67, 1341.48, 1348.66,
    1320.61],
    'Adj Close': [1105.62, 1120.84, 1097.88, 1186.92, 1186.51, 1210.28, 1211.45, 1217.56, 1269.23,
    1262.47, 1263.47, 1283.25, 1266.61, 1216.34, 1263.21, 1276.31, 1279.31, 1275.88, 1233.67, 1341.48,
    1348.66, 1320.61],
    'Volume': [2343100, 1964900, 2313400, 2664700, 2387300, 1975100, 2175400, 1739800, 2470400,
    1671700, 2518100, 1949000, 1695500, 2153000, 2093100, 1566200, 1640400, 1680600, 2951300,
    3793600, 2665400, 2072500]
  ]
  # Convert the 'Date' column to datetime format
  data['Date'] = pd.to_datetime(data['Date'], format='%d-%m-%Y')
  # Creating a DataFrame
  df = pd.DataFrame(data)
  # Filter data between two specific dates
  start_date = '2020-04-03'
  end_date = '2020-04-10'
  filtered_data = df[(df['Date'] >= start_date) & (df['Date'] <= end_date)]
  # Create a scatter plot
  plt.figure(figsize=(10, 6))
  plt.scatter(filtered_data['Date'], filtered_data['volume'], c=filtered_data['Close'], marker='o')

  plt.title('Trading Volume vs. Stock Price')
  plt.xlabel('Date')
  plt.ylabel('Volume')
  plt.colorbar(label='Close Price')

  # Show the plot
  plt.show()

```

Automatic saving failed. This file was updated remotely or in another tab. [Show diff](#)

90°F Mostly sunny

road safety management system - Colab Untitled91.ipynb - Colab Query-processing/QP DAY 1 EX | QP DAY 1 EXPERIMENTS.pdf | what is the use of xtricks - Google Search
 colab.research.google.com/drive/10PgZmOYEEKCAvH50RBQMEzGFE7HdsGOY?authuser=0#scrollTo=5InuAYDXWNQV

Untitled91.ipynb ★
 File Edit View Insert Runtime Tools Help Save failed
 + Code + Text
 {x} 4s
 # Convert the 'Date' column to datetime format
 data['Date'] = pd.to_datetime(data['Date'], format='%d-%m-%Y')
 # Creating a DataFrame
 df = pd.DataFrame(data)
 # Filter data between two specific dates
 start_date = '2020-04-03'
 end_date = '2020-04-10'
 filtered_data = df[(df['Date'] >= start_date) & (df['Date'] <= end_date)]
 # Create a scatter plot
 plt.figure(figsize=(10, 6))
 plt.scatter(filtered_data['Date'], filtered_data['volume'], c=filtered_data['Close'], marker='o')

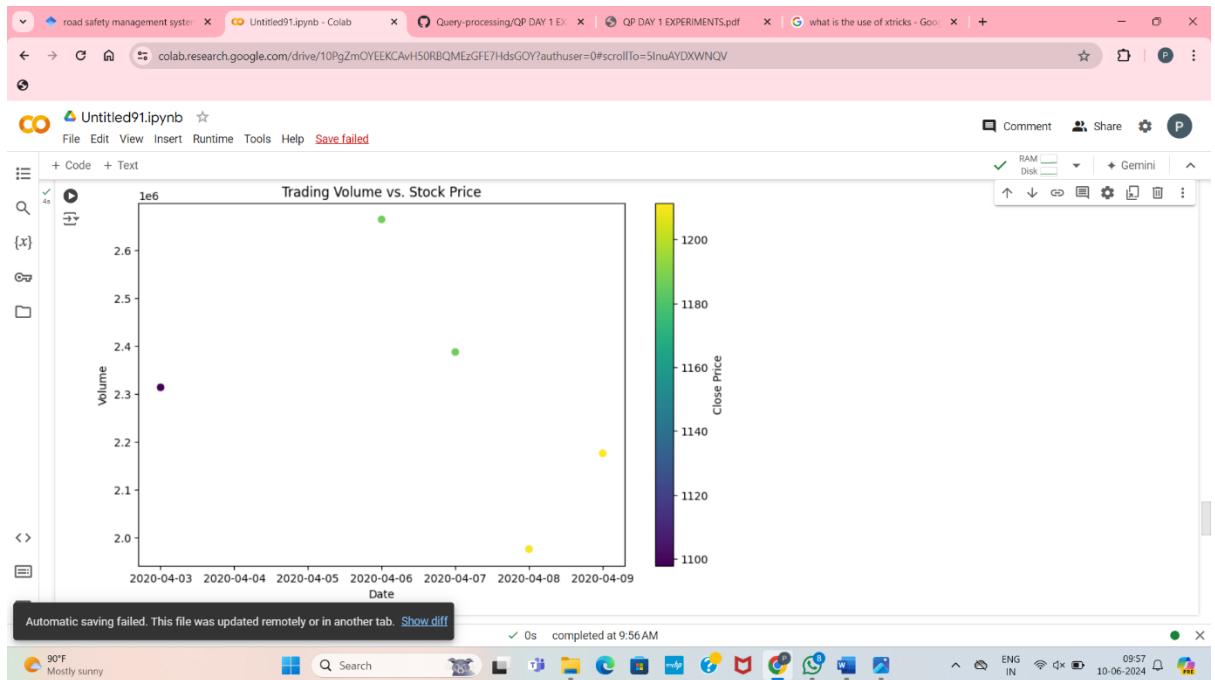
 plt.title('Trading Volume vs. Stock Price')
 plt.xlabel('Date')
 plt.ylabel('Volume')
 plt.colorbar(label='Close Price')

 # Show the plot
 plt.show()

Automatic saving failed. This file was updated remotely or in another tab. [Show diff](#)

90°F Mostly sunny

OUTPUT:



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)

CODE & OUTPUT:

```

import pandas as pd
# Sample sales data
data = [
    {'Item': 'A', 'Sale': 100},
    {'Item': 'B', 'Sale': 150},
    {'Item': 'C', 'Sale': 200},
    {'Item': 'A', 'Sale': 120},
    {'Item': 'B', 'Sale': 180},
    {'Item': 'C', 'Sale': 250},
    {'Item': 'A', 'Sale': 130},
    {'Item': 'B', 'Sale': 220},
    {'Item': 'C', 'Sale': 180}
]
# Create a DataFrame from the sample data
sales_data = pd.DataFrame(data)
# Create a pivot table to find maximum and minimum sale values for each item
pivot_table = sales_data.pivot_table(index='Item', values='Sale', aggfunc=['max', 'min'])
# Reset column names for the pivot table
pivot_table.columns = ['Max Sale', 'Min Sale']
# Display the pivot table
print(pivot_table)

```

Item	Max Sale	Min Sale
A	220	100
B	250	130
C	200	120

2. Write a Pandas program to create a Pivot table and find the item wise unit sold. .(refer sales_data table)

CODE AND OUTPUT:

The screenshot shows a Google Colab notebook titled "Untitled91.ipynb". The code cell [35] contains the following Python script:

```

[35] Item      Date      Unit Price
    A        220       100
    B        250       130
    C        180       120

import pandas as pd
# Sample sales data
data = {
    'Item': ['A', 'B', 'A', 'C', 'B', 'C', 'A', 'B', 'C'],
    'Units Sold': [10, 15, 20, 12, 25, 18, 22, 13, 16]
}
# Create a DataFrame from the sample data
sales_data = pd.DataFrame(data)
# Create a pivot table to find unit sold for each item
pivot_table = sales_data.pivot_table(index='Item', values='Units Sold', aggfunc='sum')
# Reset the column name for the pivot table
pivot_table.columns = ['Total Units Sold']
# Display the pivot table
print(pivot_table)

```

The output cell [25] shows the resulting pivot table:

Item	Total Units Sold
A	52
B	53
C	46

At the bottom of the notebook, there is a message: "Automatic saving failed. This file was updated remotely or in another tab. Show diff".

3. 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:

OrderDate	Region	Manager	SalesMan	Item	Units	Unit_price	Sale_amt
1-6-18	East	Martha	Alexander	Television	95	1,198.00	1,13,810.00
1-23-18	Central	Hermann	Shelli	Home Theater	50	500.00	25,000.00
2-9-18	Central	Hermann	Luis	Television	36	1,198.00	43,128.00
2-26-18	Central	Timothy	David	Cell Phone	27	225.00	6,075.00
3-15-18	West	Timothy	Stephen	Television	56	1,198.00	67,088.00
4-1-18	East	Martha	Alexander	Home Theater	60	500.00	30,000.00
4-18-18	Central	Martha	Steven	Television	75	1,198.00	89,850.00
5-5-18	Central	Hermann	Luis	Television	90	1,198.00	1,07,820.00
5-22-18	West	Douglas	Michael	Television	32	1,198.00	38,336.00
6-8-18	East	Martha	Alexander	Home Theater	60	500.00	30,000.00
6-25-18	Central	Hermann	Sigal	Television	90	1,198.00	1,07,820.00
7-12-18	East	Martha	Diana	Home Theater	29	500.00	14,500.00
7-29-18	East	Douglas	Karen	Home Theater	81	500.00	40,500.00

8-15-18	East	Martha	Alexander	Television	35	1,198.00	41,930.00
9-1-18	Central	Douglas	John	Desk	2	125.00	250.00
9-18-18	East	Martha	Alexander	Video Games	16	58.50	936.00
10-5-18	Central	Hermann	Sigal	Home Theater	28	500.00	14,000.00
10-22-18	East	Martha	Alexander	Cell Phone	64	225.00	14,400.00

```

Untitled91.ipynb
File Edit View Insert Runtime Tools Help Save failed
+ Code + Text
import pandas as pd
# Create a DataFrame with the provided sales data
data = [
    'OrderDate': ['1-6-18', '1-23-18', '2-9-18', '2-26-18', '3-15-18', '4-1-18', '4-18-18', '5-5-18', '5-22-18', '6-8-18',
    '6-25-18', '7-12-18', '7-29-18', '8-15-18', '9-1-18', '9-18-18', '10-5-18', '10-22-18'],
    'Region': ['East', 'Central', 'Central', 'Central', 'West', 'East', 'Central', 'Central', 'West', 'East', 'Central', 'East'],
    'Manager': ['Martha', 'Hermann', 'Hermann', 'Timothy', 'Martha', 'Martha', 'Hermann', 'Douglas',
    'Martha', 'Hermann', 'Martha', 'Douglas', 'Martha', 'Hermann', 'Martha'],
    'SalesMan': ['Alexander', 'Shelli', 'Luis', 'David', 'Stephen', 'Alexander', 'Steven', 'Luis', 'Michael', 'Alexander',
    'Sigal', 'Olana', 'Karen', 'John', 'Alexander', 'Sigal', 'Alexander'],
    'Item': ['Television', 'Home Theater', 'Television', 'Cell Phone', 'Television', 'Home Theater', 'Television',
    'Television', 'Television', 'Home Theater', 'Television', 'Home Theater', 'Television', 'Desk',
    'Video Games', 'Home Theater', 'Cell Phone'],
    'Units': [95, 50, 36, 27, 56, 60, 75, 90, 32, 60, 90, 29, 81, 35, 2, 16, 28, 64],
    'Unit_price': [1198.00, 500.00, 1198.00, 225.00, 1198.00, 500.00, 1198.00, 1198.00, 500.00,
    1198.00, 500.00, 1198.00, 125.00, 58.50, 500.00, 225.00],
    'Sale_amt': [13810.00, 25000.00, 43128.00, 6075.00, 67088.00, 30000.00, 89850.00, 107820.00, 38336.00,
    30000.00, 107820.00, 14500.00, 40500.00, 41930.00, 250.00, 936.00, 14000.00, 14400.00]
]
df = pd.DataFrame(data)
# Create a pivot table for total sale amount region-wise
pivot_region = df.pivot_table(index='Region', values='Sale_amt', aggfunc='sum')
# Create a pivot table for total sale amount manager-wise
pivot_manager = df.pivot_table(index='Manager', values='Sale_amt', aggfunc='sum')
# Create a pivot table for total sale amount salesman-wise
pivot_salesman = df.pivot_table(index='SalesMan', values='Sale_amt', aggfunc='sum')
print("Total Sale Amount Region-wise:")

```

Automatic saving failed. This file was updated remotely or in another tab. Show diff

```

30000.00, 107820.00, 14500.00, 40500.00, 41930.00, 250.00, 936.00, 14000.00, 14400.00]
}
df = pd.DataFrame(data)
# Create a pivot table for total sale amount region-wise
pivot_region = df.pivot_table(index='Region', values='Sale_amt', aggfunc='sum')
# Create a pivot table for total sale amount manager-wise
pivot_manager = df.pivot_table(index='Manager', values='Sale_amt', aggfunc='sum')
# Create a pivot table for total sale amount salesman-wise
pivot_salesman = df.pivot_table(index='SalesMan', values='Sale_amt', aggfunc='sum')
print("Total Sale Amount Region-wise:")
print(pivot_region)
print("\nTotal Sale Amount Manager-wise:")
print(pivot_manager)
print("\nTotal Sale Amount Salesman-wise:")
print(pivot_salesman)

Total Sale Amount Region-wise:
Sale_amt
Region
Central 393943.0
East 186076.0
West 105424.0

Total Sale Amount Manager-wise:
Sale_amt
Manager
Douglas 79086.0
Hermann 29768.0
Martha 235426.0
Timothy 73163.0

```

Automatic saving failed. This file was updated remotely or in another tab. Show diff

OUTPUT:

```
+ Code + Text
Total Sale Amount Region-wise:
Sale_amt
Region
Central 393943.0
East 186076.0
West 105424.0

Total Sale Amount Manager-wise:
Sale_amt
Manager
Douglas 79086.0
Hermann 29768.0
Martha 235426.0
Timothy 73163.0

Total Sale Amount Salesman-wise:
Sale_amt
SalesMan
Alexander 131076.0
David 6875.0
Diana 14590.0
John 250.0
Karen 46590.0
Luis 156948.0
Michael 38138.0
Shellie 2500.0
Sigal 121520.0
Stephen 67088.0
Steven 89850.0

Automatic saving failed. This file was updated remotely or in another tab. Show diff
```

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

CODE:

```
import pandas as pd
import numpy as np
np.random.seed(24)
df = pd.DataFrame({'A': np.linspace(1, 10, 10)})
df = pd.concat([df, pd.DataFrame(np.random.randn(10, 4), columns=list('BCDE'))], axis=1)
print("Original array:")
print(df)

def color_negative_red(val):
    color = 'red' if val < 0 else 'black'
    return 'color: %s' % color
print("\nNegative numbers red and positive numbers black:")
df.style.applymap(color_negative_red)
```

Original array:

	A	B	C	D	E
0	1.0	1.329212	-0.770033	-0.316280	-0.990810
1	2.0	-1.070816	-1.438713	0.564417	0.295722
2	3.0	-1.626404	0.219565	0.678805	1.889273
3	4.0	0.961538	0.104011	-0.481165	0.850229
4	5.0	1.453425	1.057737	0.165562	0.515018
5	6.0	-1.36936	0.562861	1.392855	-0.063328
6	7.0	0.121668	1.207603	-0.002040	1.627796
7	8.0	0.354493	1.037528	-0.385684	0.519818
8	9.0	1.686583	-1.325963	1.428984	-2.089354
9	10.0	-0.129820	0.631523	-0.586538	0.290720

Negative numbers red and positive numbers black:

	A	B	C	D	E
0	1.000000	1.329212	-0.770033	-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	-0.481165	0.850229
4	5.000000	1.453425	1.057737	0.165562	0.515018
5	6.000000	-1.36936	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	0.290720

OUTPUT:

```
import pandas as pd
import numpy as np
np.random.seed(24)
df = pd.DataFrame({'A': np.linspace(1, 10, 10)})
df = pd.concat([df, pd.DataFrame(np.random.randn(10, 4), columns=list('BCDE'))], axis=1)
print("Original array:")
print(df)

def color_negative_red(val):
    color = 'red' if val < 0 else 'black'
    return 'color: %s' % color
print("\nNegative numbers red and positive numbers black:")
df.style.applymap(color_negative_red)
```

Original array:

	A	B	C	D	E
0	1.0	1.329212	-0.770033	-0.316280	-0.990810
1	2.0	-1.070816	-1.438713	0.564417	0.295722
2	3.0	-1.626404	0.219565	0.678805	1.889273
3	4.0	0.961538	0.104011	-0.481165	0.850229
4	5.0	1.453425	1.057737	0.165562	0.515018
5	6.0	-1.36936	0.562861	1.392855	-0.063328
6	7.0	0.121668	1.207603	-0.002040	1.627796
7	8.0	0.354493	1.037528	-0.385684	0.519818
8	9.0	1.686583	-1.325963	1.428984	-2.089354
9	10.0	-0.129820	0.631523	-0.586538	0.290720

Negative numbers red and positive numbers black:

	A	B	C	D	E
0	1.000000	1.329212	-0.770033	-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	-0.481165	0.850229
4	5.000000	1.453425	1.057737	0.165562	0.515018
5	6.000000	-1.36936	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	0.290720

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

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

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

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	--

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

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

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
S5	s002	V	Gino McNeill	11/05/2002	14	151	31	street2
S6	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.

	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

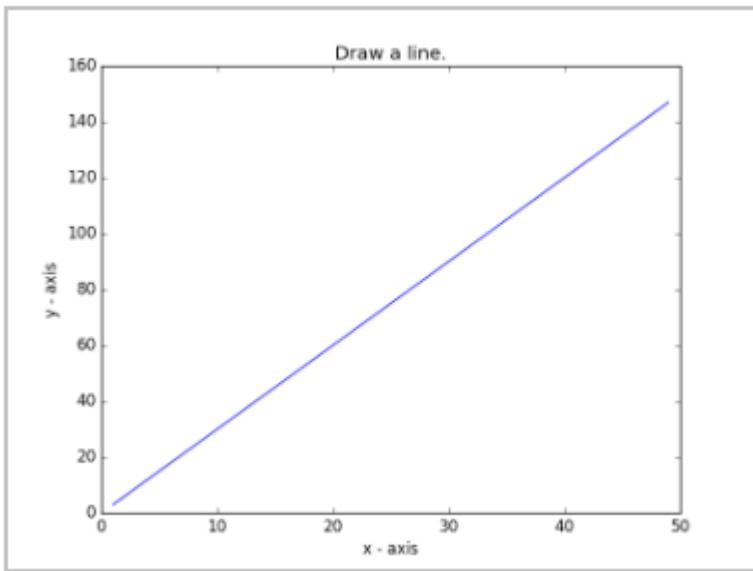
19. Write a Pandas program to display the dimensions or shape of the World alcohol consumption dataset. Also extract the column names from the dataset.

	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

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

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

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



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.

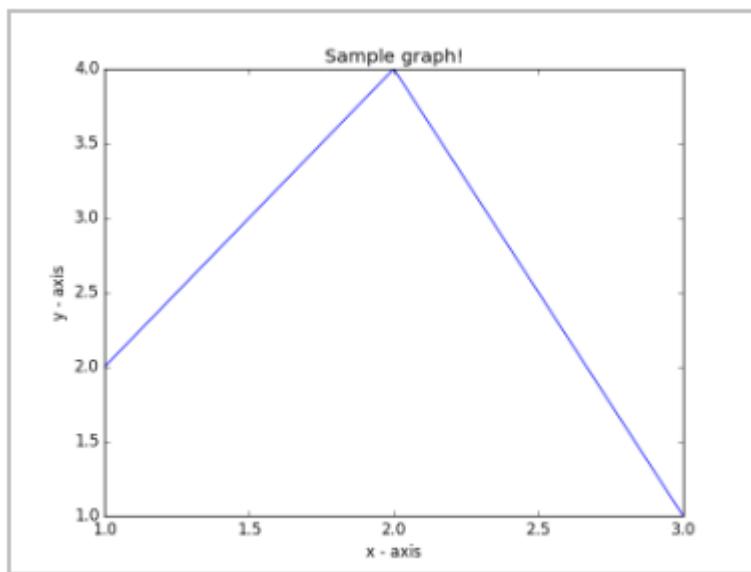
Test Data:

test.txt

1 2

2 4

3 1

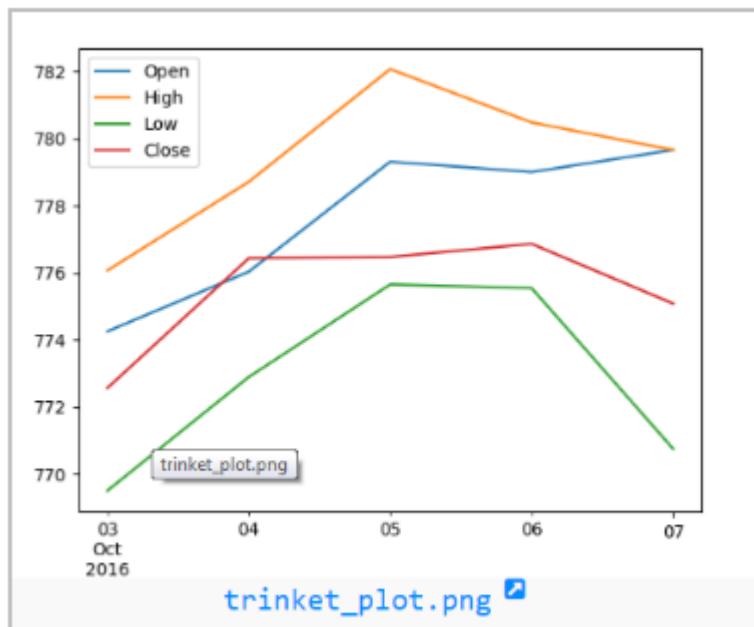


24. Write a Python program to draw line charts of the financial data of Alphabet Inc. between October 3, 2016 to October 7, 2016.

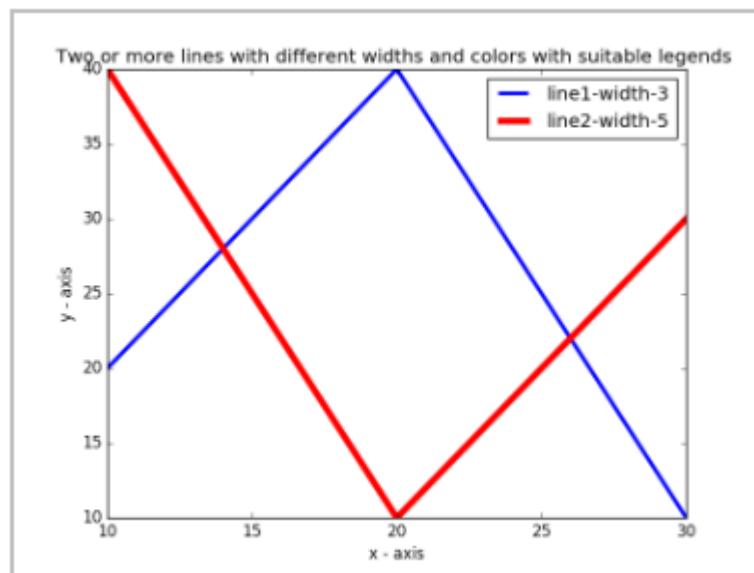
Sample Financial data (fdata.csv):

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
10-06-16,779,780.479998,775.539978,776.859985
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.



26. Write a Python program to create multiple plots.

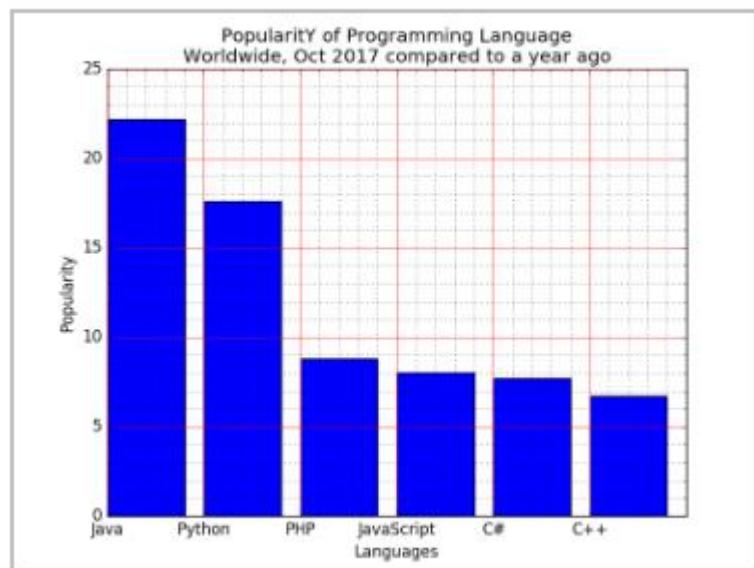


27. Write a Python programming to display a bar chart of the popularity of programming Languages.

Sample data:

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

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

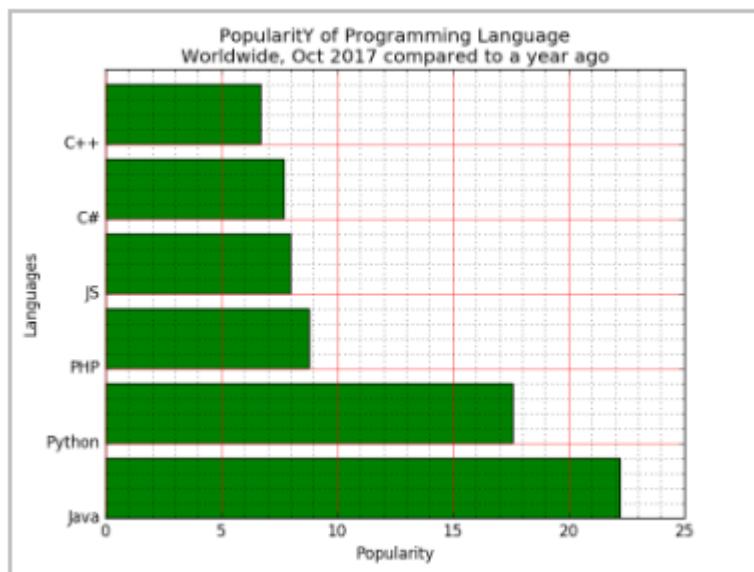


28. Write a Python programming to display a horizontal bar chart of the popularity of programming Languages.

Sample data:

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

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

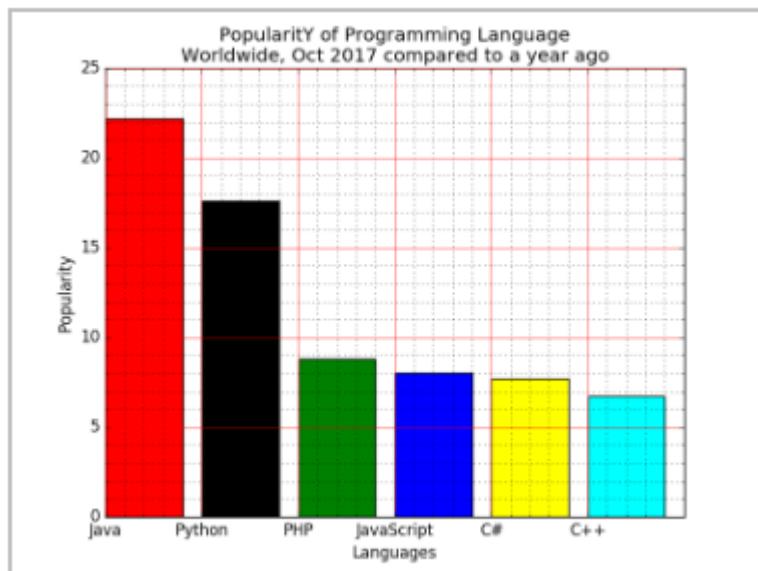


29. Write a Python programming to display a bar chart of the popularity of programming Languages. Use different color for each bar.

Sample data:

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

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

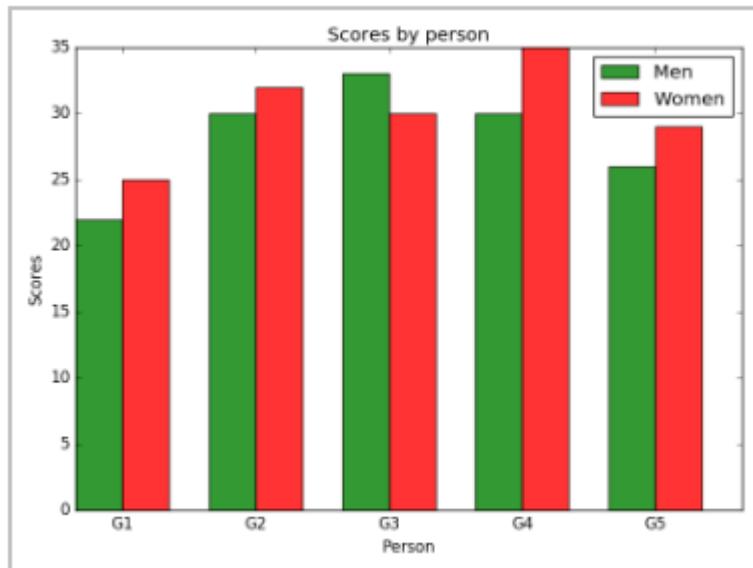


30. Write a Python program to create bar plot of scores by group and gender.
Use multiple X values on the same chart for men and women.

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.

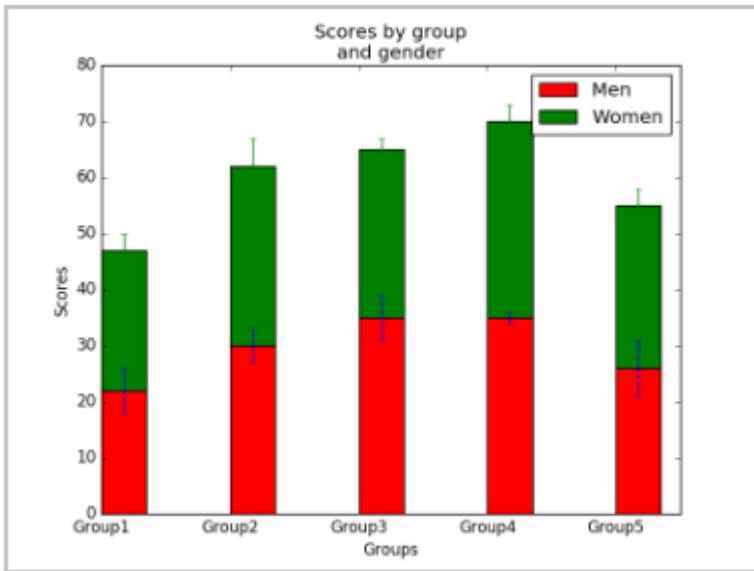
Sample Data:

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

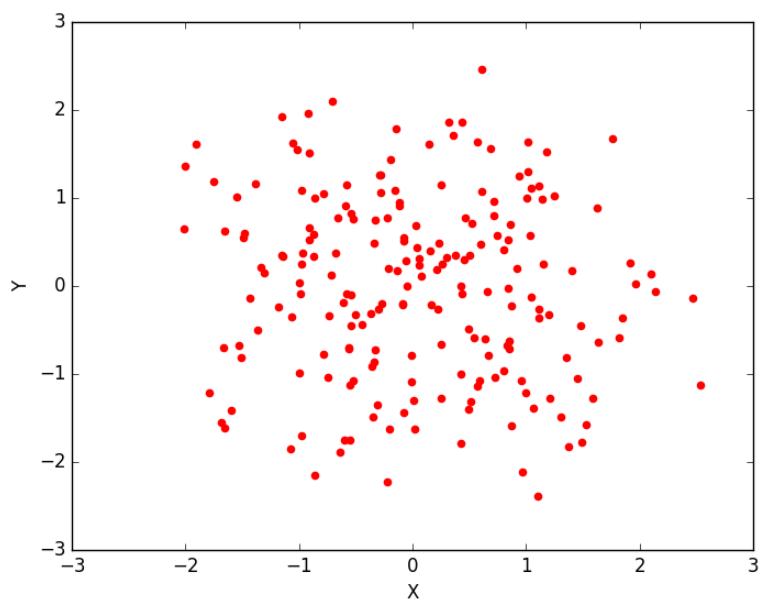
Means (women) = (25, 32, 30, 35, 29)

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

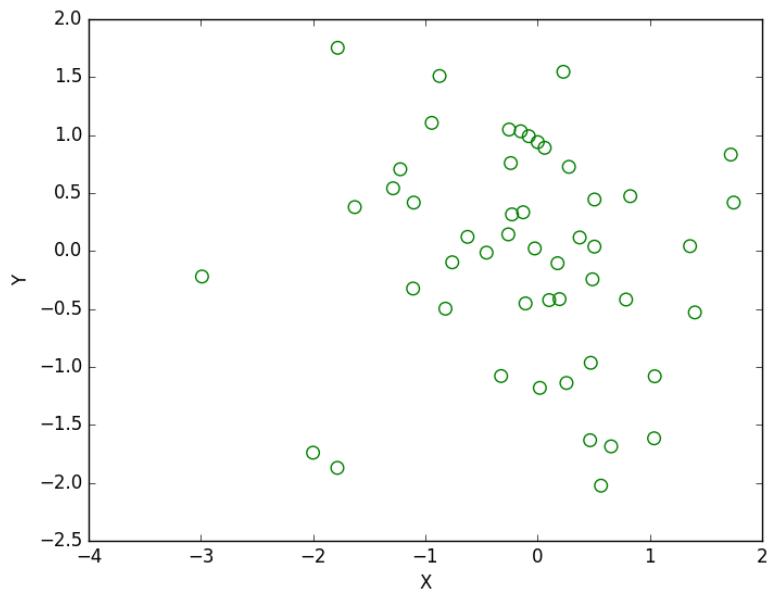
Women Standard deviation = (3, 5, 2, 3, 3)



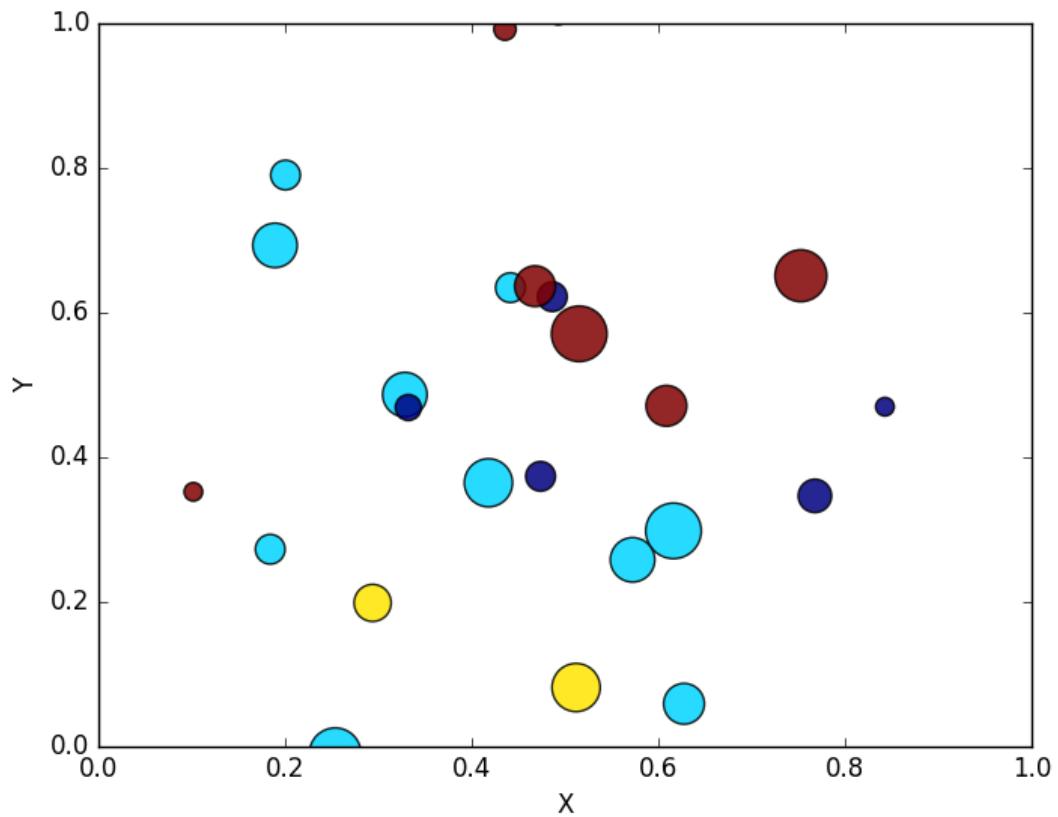
32. Write a Python program to draw a scatter graph taking a random distribution in X and Y and plotted against each other.



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.

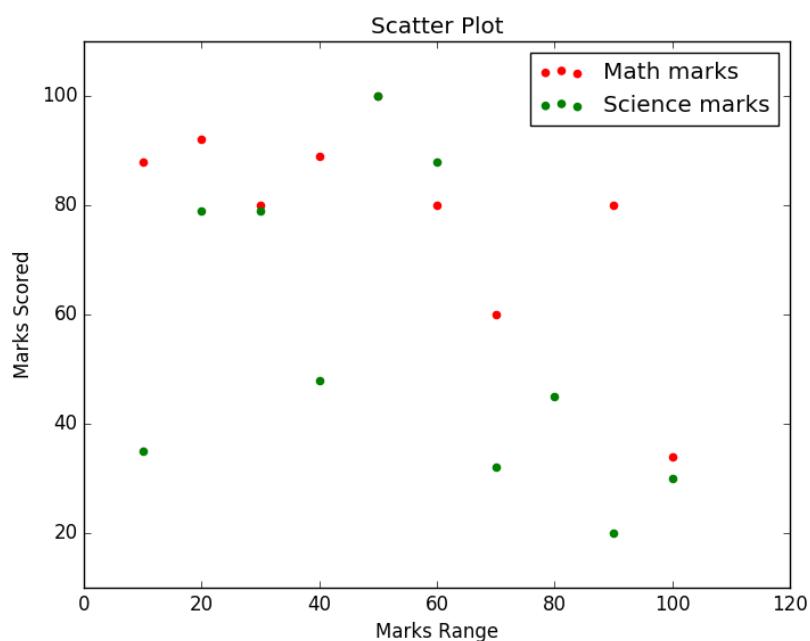
Sample data:

Test Data:

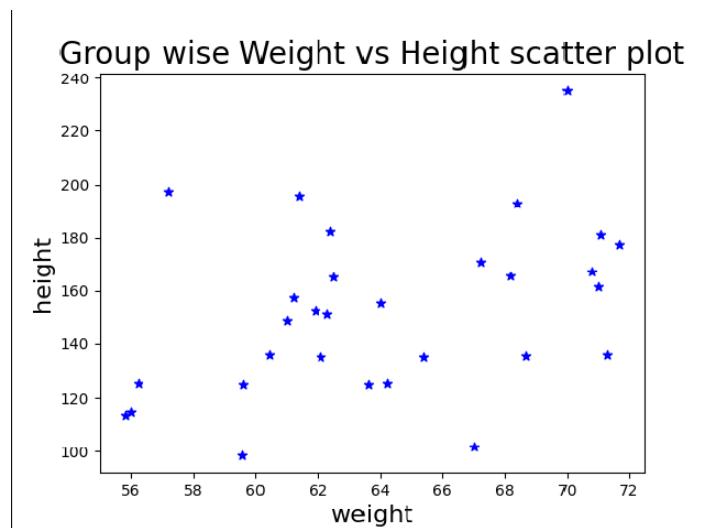
```
math_marks = [88, 92, 80, 89, 100, 80, 60, 100, 80, 34]
```

```
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]}

Expected 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.

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']
```

Expected Output:

	attempts	name	qualify	score
a	1	Anastasia	yes	12.5
b	3	Dima	no	9.0
...				
i	2	Kevin	no	8.0
j	1	Jonas	yes	19.0

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']
```

Expected Output:

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

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']
```

Expected Output:

Select specific columns:

```
   name  score  
a Anastasia  12.5  
b      Dima  9.0  
c Katherine  16.5  
...  
h      Laura  NaN  
i      Kevin  8.0  
j      Jonas  19.0
```