

DSA0508 Query Processing for Data Science with data exploration

LAB QUESTIONS

SRI RAM P

192224162

1. Write a Pandas program to select distinct department id from employee's file.

CODE:

```
import pandas as pd

data = {

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

    'DEPARTMENT_NAME': ['Administration', 'Marketing', 'Purchasing', 'Human
Resources', 'Shipping', 'IT', 'Public Relations', 'Sales', 'Executive', 'Finance', 'Accounting',
'Treasury', 'Corporate Tax', 'Control And Credit', 'Shareholder Services', 'Benefits',
'Manufacturing', 'Construction', 'Contracting', 'Operations', 'IT Support', 'NOC', 'IT
Helpdesk', 'Government Sales', 'Retail Sales', 'Recruiting', 'Payroll'], 

    'MANAGER_ID': [200, 201, 114, 203, 121, 103, 204, 145, 100, 108, 205, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], 

    'LOCATION_ID': [1700, 1800, 1700, 2400, 1500, 1400, 2700, 2500, 1700, 1700, 1700,
1700, 1700, 1700, 1700, 1700, 1700, 1700, 1700, 1700, 1700, 1700, 1700, 1700, 1700,
1700, 1700]

}

# Convert the dictionary into a DataFrame

df = pd.DataFrame(data)

# Select distinct department IDs

distinct_department_ids = df['DEPARTMENT_ID'].unique()

# Print the distinct department IDs

print("Distinct Department IDs:")

print(distinct_department_ids)
```

```

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

import pandas as pd

data = {
    'DEPARTMENT_ID': [10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120, 130, 140, 150, 160, 170, 180, 190, 200, 210, 220, 230, 240, 250, 260, 270],
    'DEPARTMENT_NAME': ['Administration', 'Marketing', 'Purchasing', 'Human Resources', 'Shipping', 'IT', 'Public Relations', 'Sales', 'Executive', 'Finance'],
    'MANAGER_ID': [200, 201, 114, 203, 121, 103, 204, 145, 100, 108, 205, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'LOCATION_ID': [1700, 1800, 1700, 2400, 1500, 1400, 2700, 1700, 1700, 1700, 1700, 1700, 1700, 1700, 1700, 1700, 1700, 1700, 1700]
}

# Convert the dictionary into a DataFrame
df = pd.DataFrame(data)

# Select distinct department IDs
distinct_department_ids = df['DEPARTMENT_ID'].unique()

# Print the distinct department IDs
print("Distinct Department IDs:")
print(distinct_department_ids)

```

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

CODE:

```

import pandas as pd

data = {

    'EMPLOYEE_ID': [102, 101, 101, 201, 114, 122, 200, 176, 176, 200], 

    'START_DATE': ['2001-01-13', '1997-09-21', '2001-10-28', '2004-02-17', '2006-03-24', 
    '2007-01-01', '1995-09-17', '2006-03-24', '2007-01-01', '2002-07-01'], 

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

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

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

}

# Convert the dictionary into a DataFrame

df = pd.DataFrame(data)

# Group by employee ID and count the number of unique job IDs

employee_job_counts = df.groupby('EMPLOYEE_ID')['JOB_ID'].nunique()

# Filter employees who did two or more jobs in the past

employees_with_multiple_jobs = employee_job_counts[employee_job_counts >= 2]

```

Print the employee IDs who did two or more jobs in the past

```
print("Employee IDs who did two or more jobs in the past:")  
print(employees_with_multiple_jobs.index.tolist())
```

The screenshot shows a code editor window titled 'LAB2.py' and an 'IDLE Shell 3.12.0' window. The code in the editor is a Python script that reads data from a CSV file, groups it by employee ID, filters employees with multiple jobs, and prints their IDs. The shell window shows the script being run and the output of the printed list.

```
File Edit Format Run Options Window Help  
# Write a Pandas program to display the ID for those employees who did two or more jobs in the past  
  
import pandas as pd  
  
data = {  
    'EMPLOYEE_ID': [102, 101, 101, 201, 114, 122, 200, 176, 176, 200],  
    'START_DATE': ['2001-01-13', '1997-09-21', '2001-10-28', '2004-02-17', '2006-03-24', '2007-01-01', '1995-09-17', '2006-03-24', '2007-01-01', '2002-07-01'],  
    'END_DATE': ['2006-07-24', '2001-10-27', '2005-03-15', '2007-12-19', '2007-12-31', '2007-12-31', '2001-06-17', '2006-12-31', '2007-12-31', '2006-12-31'],  
    'JOB_ID': ['IT_PROG', 'AC_ACCOUNT', 'AC_MGR', 'MK_REP', 'ST_CLERK', 'AD_ASST', 'SA REP', 'SA_MAN', 'AC_ACCOUNT'],  
    'DEPARTMENT_ID': [60, 110, 110, 20, 50, 50, 90, 90, 90]  
}  
  
# Convert the dictionary into a DataFrame  
df = pd.DataFrame(data)  
  
# Group by employee ID and count the number of unique job IDs  
employee_job_counts = df.groupby('EMPLOYEE_ID')['JOB_ID'].nunique()  
  
# Filter employees who did two or more jobs in the past  
employees_with_multiple_jobs = employee_job_counts[employee_job_counts >= 2]  
  
# Print the employee IDs who did two or more jobs in the past  
print("Employee IDs who did two or more jobs in the past:")  
print(employees_with_multiple_jobs.index.tolist())
```

```
File Edit Shell Debug Options Window Help  
Python 3.12.0 (tags/v3.12.0:0fb18b0, Oct 2 2023, 13:03:39) [MSC v.1935 64 bit (AMD64)] on win32  
Type "help", "copyright", "credits" or "license()" for more information.  
=>>>  
= RESTART: D:\Query processing\LAB QUESTION\LAB2.py  
Employee IDs who did two or more jobs in the past:  
[101, 176, 200]
```

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

CODE:

```
import pandas as pd  
  
data = {  
  
    'JOB_ID': ['AD_PRES', 'AD_VP', 'AD_ASST', 'FI_MGR', 'FI_ACCOUNT',  
    'AC_MGR', 'AC_ACCOUNT', 'SA_MAN', 'SA REP', 'PU_MAN', 'PU_CLERK',  
    'ST_MAN', 'ST_CLERK', 'SH_CLERK', 'IT_PROG', 'MK_MAN', 'MK_REP', 'HR_REP',  
    'PR_REP'],  
  
    'JOB_TITLE': ['President', 'Administration Vice President', 'Administration Assistant',  
    'Finance Manager', 'Accountant', 'Accounting Manager', 'Public Accountant', 'Sales  
    Manager', 'Sales Representative', 'Purchasing Manager', 'Purchasing Clerk', 'Stock  
    Manager', 'Stock Clerk', 'Shipping Clerk', 'Programmer', 'Marketing Manager', 'Marketing  
    Representative', 'Human Resources Representative', 'Public Relations Representative'],  
  
    'MIN_SALARY': [20080, 15000, 3000, 8200, 4200, 8200, 4200, 10000, 6000, 8000,  
    2500, 5500, 2008, 2500, 4000, 9000, 4000, 4000, 4500],  
  
    'MAX_SALARY': [40000, 30000, 6000, 16000, 9000, 16000, 9000, 20080, 12008,  
    15000, 5500, 8500, 5000, 5500, 10000, 15000, 9000, 9000, 10500]  
  
}  
  
df = pd.DataFrame(data)
```

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

```
LAB3.py - D:\Query processing\LAB QUESTION\LAB3.py (3.12.0)
File Edit Format Run Options Window Help
# Write a Pandas program to display the details of jobs in descending sequence on job title.

import pandas as pd

# Create the DataFrame from the given 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',
    'JOB_TITLE': ['President', 'Administration Vice President', 'Administration Assistant', 'Finance Manager', 'Accountant', 'Accounting Manager', 'Public Ac
    'MIN_SALARY': [20080, 15000, 3000, 8200, 4200, 8200, 4200, 10000, 6000, 8000, 2500, 5500, 2008, 2500, 4000, 9000, 4000, 4000, 4500],
    'MAX_SALARY': [40000, 30000, 6000, 16000, 9000, 16000, 20080, 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)

# Display the sorted DataFrame
print(df_sorted)
```

```
IDLE Shell 3.12.0
File Edit Shell Debug Options Window Help
Python 3.12.0 (tags/v3.12.0:fdb18b0, Oct 2 2023, 13:03:39) [MSC v.1935 64 bit (AMD64)]
4] on win32
Type "help", "copyright", "credits" or "license()" for more information.

>>> = RESTART: D:\Query processing\LAB QUESTION\LAB3.py
      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 Rep.    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 Rep.     4000      10000
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.

CODE:

```
import pandas as pd

import matplotlib.pyplot as plt

import numpy as np

start_date = '2024-01-01'

end_date = '2024-02-21'

num_dates = 21

date_range = pd.date_range(start=start_date, end=end_date, periods=num_dates)

prices = [100, 110, 105, 115, 120, 118, 125, 130, 135, 140, 145, 150, 155, 160, 165, 170,
175, 180, 185, 190, 195]

stock_data = pd.DataFrame({'Date': date_range, 'Price': prices})

stock_data.plot(kind='line', x='Date', y='Price', figsize=(10, 6), color='blue')

plt.title('Historical Stock Prices of Alphabet Inc.')

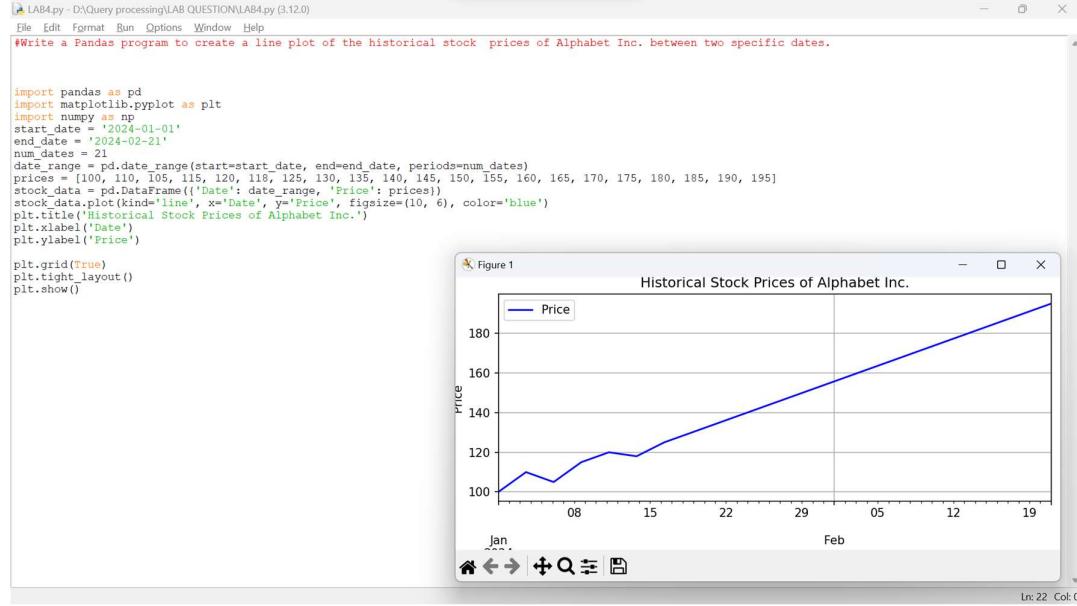
plt.xlabel('Date')

plt.ylabel('Price')
```

```

plt.grid(True)
plt.tight_layout()
plt.show()

```



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

CODE:

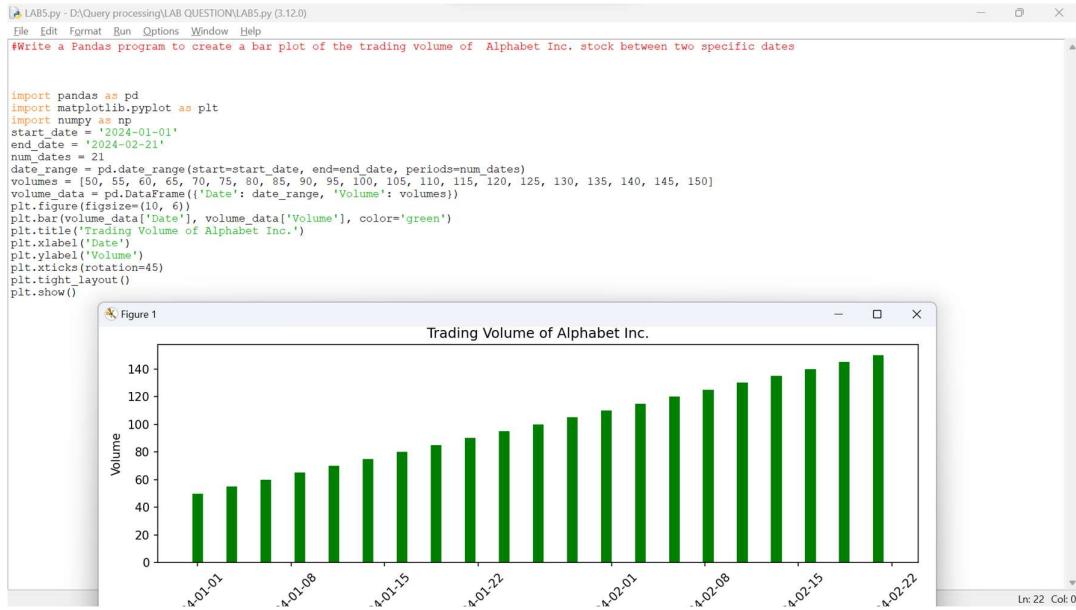
```

import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
start_date = '2024-01-01'
end_date = '2024-02-21'
num_dates = 21
date_range = pd.date_range(start=start_date, end=end_date, periods=num_dates)
volumes = [50, 55, 60, 65, 70, 75, 80, 85, 90, 95, 100, 105, 110, 115, 120, 125, 130, 135, 140, 145, 150]
volume_data = pd.DataFrame({'Date': date_range, 'Volume': volumes})
plt.figure(figsize=(10, 6))
plt.bar(volume_data['Date'], volume_data['Volume'], color='green')
plt.title('Trading Volume of Alphabet Inc.')
plt.xlabel('Date')
plt.ylabel('Volume')
plt.xticks(rotation=45)

```

```
plt.tight_layout()
```

```
plt.show()
```



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

CODE:

```
import pandas as pd

import matplotlib.pyplot as plt

start_date = '2024-01-01'

end_date = '2024-02-21'

num_dates = 21

date_range = pd.date_range(start=start_date, end=end_date, periods=num_dates)

prices = [100, 110, 105, 115, 120, 118, 125, 130, 135, 140, 145, 150, 155, 160, 165, 170, 175, 180, 185, 190, 195]

volumes = [50, 55, 60, 65, 70, 75, 80, 85, 90, 95, 100, 105, 110, 115, 120, 125, 130, 135, 140, 145, 150]

stock_data = pd.DataFrame({'Date': date_range, 'Price': prices, 'Volume': volumes})

start_date = '2024-01-10'

end_date = '2024-02-10'

filtered_data = stock_data[(stock_data['Date'] >= start_date) & (stock_data['Date'] <= end_date)]

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

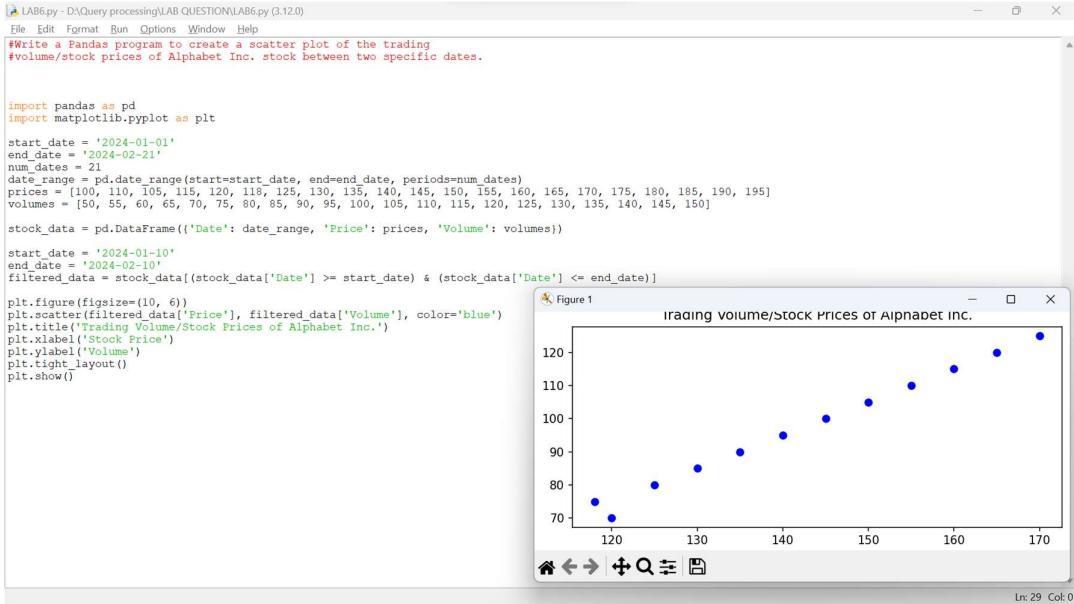
plt.scatter(filtered_data['Price'], filtered_data['Volume'], color='blue')

plt.title('Trading Volume/Stock Prices of Alphabet Inc.')
```

```

plt.xlabel('Stock Price')
plt.ylabel('Volume')
plt.tight_layout()
plt.show()

```



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

CODE:

```

import pandas as pd
data = {
    'Item': ['A', 'B', 'A', 'B', 'A', 'B'],
    'Date': ['2023-01-01', '2023-01-01', '2023-01-02', '2023-01-02', '2023-01-03', '2023-01-03'],
    'Sale': [100, 150, 120, 130, 110, 160]
}
sales_data = pd.DataFrame(data)
sales_data['Date'] = pd.to_datetime(sales_data['Date'])
pivot_table = pd.pivot_table(sales_data, values='Sale', index='Item', aggfunc=['max', 'min'])
pivot_table.columns = ['Max Sale', 'Min Sale']
print("Pivot Table:")
print(pivot_table)
overall_max_sale = sales_data['Sale'].max()

```

```

overall_min_sale = sales_data['Sale'].min()

print("\nOverall Maximum Sale Value:", overall_max_sale)

print("Overall Minimum Sale Value:", overall_min_sale)

```

```

#LAB7.py - D:\Query processing\LAB QUESTION\LAB7.py (3.12.0)
File Edit Format Run Options Window Help
#Write a Pandas program to create a Pivot table and find the maximum and minimum sale value of the items.

import pandas as pd
data = {
    'Item': ['A', 'B', 'A', 'B', 'A', 'B'],
    'Date': ['2023-01-01', '2023-01-01', '2023-01-02', '2023-01-02', '2023-01-03', '2023-01-03'],
    'Sale': [100, 150, 120, 130, 110, 160]
}
sales_data = pd.DataFrame(data)
sales_data['Date'] = pd.to_datetime(sales_data['Date'])
pivot_table = pd.pivot_table(sales_data, values='Sale', index='Item', aggfunc=['max', 'min'])
pivot_table.columns = ['Max Sale', 'Min Sale']
print("Pivot Table:")
print(pivot_table)
overall_max_sale = sales_data['Sale'].max()
overall_min_sale = sales_data['Sale'].min()
print("\nOverall Maximum Sale Value:", overall_max_sale)
print("Overall Minimum Sale Value:", overall_min_sale)

```

IDLE Shell 3.12.0

```

File Edit Shell Debug Options Window Help
Python 3.12.0 (tags/v3.12.0:0fb18b0, Oct 2 2023, 13:03:39) [MSC v.1935 64 bit (AMD64)] on Win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: D:\Query processing\LAB QUESTION\LAB7.py
=====
===== RESTART: D:\Query processing\LAB QUESTION\LAB7.py =====
=====
Pivot Table:
 Max Sale Min Sale
Item
A      120     100
B      160     130

Overall Maximum Sale Value: 160
Overall Minimum Sale Value: 100

```

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

CODE:

```

import pandas as pd

data = {

    'Item': ['A', 'B', 'A', 'B', 'A', 'B'],

    'Date': ['2023-01-01', '2023-01-01', '2023-01-02', '2023-01-02', '2023-01-03', '2023-01-03'],

    'Unit Sold': [10, 15, 12, 13, 11, 16]

}

sales_data = pd.DataFrame(data)

sales_data['Date'] = pd.to_datetime(sales_data['Date'])

pivot_table = pd.pivot_table(sales_data, values='Unit Sold', index='Item', aggfunc='sum')

print("Item-wise Unit Sold Pivot Table:")

print(pivot_table)

```

```

LAB8.py - D:\Query processing\LAB QUESTION\LAB8.py (3.12.0)
File Edit Format Run Options Window Help
#Write a Pandas program to create a Pivot table and find the item wise unit sold

import pandas as pd
data = {
    'Item': ['A', 'B', 'A', 'B', 'A', 'B'],
    'Date': ['2023-01-01', '2023-01-02', '2023-01-02', '2023-01-02', '2023-01-03', '2023-01-03'],
    'Unit Sold': [10, 15, 12, 13, 11, 16]
}
sales_data = pd.DataFrame(data)
sales_data['Date'] = pd.to_datetime(sales_data['Date'])
pivot_table = pd.pivot_table(sales_data, values='Unit Sold', index='Item', aggfunc='sum')
print("Item-wise Unit Sold Pivot Table:")
print(pivot_table)

```

```

IDLE Shell 3.12.0
File Edit Shell Debug Options Window Help
Python 3.12.0 (tags/v3.12.0:0fb18b0, Oct 2 2023, 13:03:39) [MSC v.1935 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: D:\Query processing\LAB QUESTION\LAB8.py
Item-wise Unit Sold Pivot Table:
   Item      Unit Sold
   A          33
   B          44
>>>

```

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

CODE:

```

import pandas as pd

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', 'East', 'Central', 'East', 'Central', 'Central', 'East'],

    'Manager': ['Martha', 'Hermann', 'Hermann', 'Timothy', 'Timothy', 'Martha', 'Martha', 'Hermann', 'Douglas', 'Martha', 'Hermann', 'Martha', 'Douglas', 'Martha', 'Douglas', 'Martha', 'Hermann', 'Martha'],

    'SalesMan': ['Alexander', 'Shelli', 'Luis', 'David', 'Stephen', 'Alexander', 'Steven', 'Luis', 'Michael', 'Alexander', 'Sigal', 'Diana', 'Karen', 'Alexander', 'John', 'Alexander', 'Sigal', 'Alexander'],

    'Item': ['Television', 'Home Theater', 'Television', 'Cell Phone', 'Television', 'Home Theater', 'Television', 'Television', 'Television', 'Home Theater', 'Television', 'Home Theater', 'Home Theater', '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, 1198.00, 500.00, 1198.00, 500.00, 500.00, 1198.00, 125.00, 58.50, 500.00, 225.00], 

    'Sale_amt': [113810.00, 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]
}

```

```

}

sales_data = pd.DataFrame(data)

sales_data['OrderDate'] = pd.to_datetime(sales_data['OrderDate'], format='%m-%d-%y')

pivot_table = pd.pivot_table(sales_data, values='Sale_amt', index=['Region', 'Manager', 'SalesMan'], aggfunc='sum')

print("Pivot Table - Total Sale Amount:")

print(pivot_table)

```

The screenshot shows a Windows desktop with two open windows. The top window is titled 'LAB9.py - D:\Query processing\LAB QUESTION\LAB9.py (3.12.0)' and contains Python code for generating a pivot table. The bottom window is titled 'IDLE Shell 3.12.0' and displays the resulting pivot table output.

```

# Write a Pandas program to create a Pivot table and find the total sale amount region wise, manager wise, sales man wise.

import pandas as pd

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'],
    'Region': ['East', 'Central', 'Central', 'West', 'East', 'Central', 'Central', 'West', 'East', 'Central', 'East', 'East', 'Central'],
    'Manager': ['Martha', 'Hermann', 'Hermann', 'Timothy', 'Martha', 'Hermann', 'Hermann', 'Douglas', 'Martha', 'Hermann', 'Martha', 'Douglas', 'Martha'],
    'SalesMan': ['Alexander', 'Shelli', 'Luis', 'David', 'Stephen', 'Alexander', 'Steven', 'Luis', 'Michael', 'Alexander', 'Sigal', 'Diana', 'Karen', 'Alex'],
    'Item': ['Television', 'Home Theater', 'Television', 'Cell Phone', 'Television', 'Home Theater', 'Television', 'Television', 'Television', 'Television', 'Home Theater'],
    '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, 500.00, 1198.00, 500.00, 1198.00, 125.00, 58.50, 500],
    'Sale_amt': [113810.00, 25000.00, 43128.00, 6075.00, 30000.00, 89850.00, 107820.00, 38336.00, 30000.00, 107820.00, 14500.00, 40500.00, 41930.00]
]

sales_data = pd.DataFrame(data)
sales_data['OrderDate'] = pd.to_datetime(sales_data['OrderDate'], format='%m-%d-%y')

pivot_table = pd.pivot_table(sales_data, values='Sale_amt', index=['Region', 'Manager', 'SalesMan'], aggfunc='sum')

print("Pivot Table - Total Sale Amount:")
print(pivot_table)

```

Region	Manager	SalesMan	Sale_amt
Central	Douglas	John	250.0
	Hermann	Luis	150948.0
		Shelli	25000.0
		Sigal	121820.0
East	Martha	Steven	89850.0
	Timothy	David	6075.0
	Douglas	Karen	40500.0
	Martha	Alexander	231076.0
		Diana	14500.0
West	Douglas	Michael	38336.0
	Timothy	Stephen	67088.0

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.

CODE:

```

import pandas as pd

import numpy as n

np.random.seed(42)

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

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

# Define a function to apply color based on the sign of the number

def color_negative_red(val):

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

    return f<span style="color: {color};">{val:.3f}</span>

```

Apply the style to the DataFrame

```
styled_df = df.applymap(color_negative_red)
```

Convert the styled DataFrame to HTML

```

styled_html = styled_df.to_html(escape=False, index=False)

# Save the HTML output to a file

with open('styled_table.html', 'w') as f:

    f.write(styled_html)

print("HTML table saved to 'styled_table.html'")

```

OUTPUT:

A	B	C	D
0.497	-0.138	0.648	1.523
-0.234	-0.234	1.579	0.767
-0.469	0.543	-0.463	-0.466
0.242	-1.913	-1.725	-0.562
-1.013	0.314	-0.908	-1.412
1.466	-0.226	0.068	-1.425
-0.544	0.111	-1.151	0.376
-0.601	-0.292	-0.602	1.852
-0.013	-1.058	0.823	-1.221
0.209	-1.960	-1.328	0.197

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

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)

# Introducing NaN values

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

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

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

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

# Function to highlight NaN values in red

def highlight_null(val):

    color = 'red' if pd.isnull(val) else 'black'

```

```

    return 'color: %s' % color

# Apply styling to DataFrame

styled_df = df.style.applymap(highlight_null)

# Save styled DataFrame to an HTML file

styled_df.to_html('styled_dataframe.html', na_rep='')

print("HTML output saved to 'styled_dataframe.html'")

```

OUTPUT:

	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.

CODE:

```

import pandas as pd

import numpy as np

# Create a DataFrame with ten rows and four columns with random values

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

columns = ['Column1', 'Column2', 'Column3', 'Column4']

df = pd.DataFrame(data, columns=columns)

# Convert the DataFrame to strings with HTML-style formatting

def html_style(x):

    return f<span style="background-color:black; color:yellow;">{x:.4f}</span>

formatted_df = df.applymap(html_style)

# Save the HTML-formatted DataFrame to a file

html_file = 'formatted_df.html'

formatted_df.to_html(html_file, escape=False)

# Open the HTML file in the default web browser

```

```
import webbrowser  
webbrowser.open(html_file)
```

OUTPUT:

	Column1	Column2	Column3	Column4
0	0.2953	0.4267	0.1210	0.8683
1	0.0667	0.8619	0.8847	0.3478
2	0.1344	0.0886	0.6610	0.0741
3	0.4103	0.8144	0.3893	0.9763
4	0.1949	0.3023	0.9131	0.0204
5	0.1031	0.5901	0.0189	0.6667
6	0.3881	0.3465	0.7506	0.1004
7	0.4405	0.7628	0.8371	0.5164
8	0.8372	0.6094	0.6127	0.6558
9	0.0641	0.5441	0.1576	0.6028

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

CODE:

```
import pandas as pd  
import numpy as np  
  
# Creating the DataFrame with missing values  
  
data = {'ord_no': [70001.0, np.nan, 70002.0, 70004.0, np.nan, 70005.0, np.nan, 70010.0,  
70003.0, 70012.0, np.nan, 70013.0],  
  
'purch_amt': [150.50, 270.65, 65.26, 110.50, 948.50, 2400.60, 5760.00, 1983.43,  
2480.40, 250.45, 75.29, 3045.60],  
  
'ord_date': ['2012-10-05', '2012-09-10', np.nan, '2012-08-17', '2012-09-10', '2012-07-  
27', '2012-09-10', '2012-10-10', '2012-10-10', '2012-06-27', '2012-08-17', '2012-04-25'],  
  
'customer_id': [3002, 3001, 3001, 3003, 3002, 3001, 3001, 3004, 3003, 3002, 3001,  
3001],  
  
'salesman_id': [5002.0, 5003.0, 5001.0, np.nan, 5002.0, 5001.0, 5001.0, np.nan,  
5003.0, 5002.0, 5003.0, np.nan]}  
  
df = pd.DataFrame(data)  
  
# Detect missing values and display True or False  
  
missing_values = df.isnull()  
  
print(missing_values)
```

```

import pandas as pd
import numpy as np

# Creating the DataFrame with missing values
data = {'ord_no': [70001.0, np.nan, 70002.0, 70004.0, np.nan, 70005.0, np.nan, 70010.0, 70003.0, 70012.0, np.nan, 70013.0],
        'purch_amt': [150.50, 270.65, 65.26, 110.50, 948.50, 2400.60, 5760.00, 1983.43, 2480.40, 250.45, 75.29, 3045.60],
        'ord_date': ['2012-10-05', '2012-09-10', np.nan, '2012-08-17', '2012-09-10', '2012-07-27', '2012-09-10', '2012-10-10', '2012-10-10', '2012-06-27', '2012-04-25'],
        'customer_id': [3002, 3001, 3001, 3003, 3002, 3001, 3001, 3004, 3003, 3002, 3001],
        'salesman_id': [5002.0, 5003.0, 5001.0, np.nan, 5002.0, 5001.0, 5001.0, np.nan, 5003.0, 5002.0, 5003.0, np.nan]}

df = pd.DataFrame(data)

# Detect missing values and display True or False
missing_values = df.isnull()

# Display the result
print(missing_values)

```

IDLE Shell 3.12.0

```

File Edit Shell Debug Options Window Help
Python 3.12.0 (tags/v3.12.0:0fb1b8b0, Oct 2 2023, 13:03:39) [MSC v.1935 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.

>>> = RESTART: D:\Query processing\LAB QUESTION\LAB13.py
   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.

CODING:

```

import pandas as pd

import numpy as np

data = {'ord_no': [70001.0, np.nan, 70002.0, 70004.0, np.nan, 70005.0, np.nan, 70010.0, 70003.0, 70012.0, np.nan, 70013.0], 

        'purch_amt': [150.50, 270.65, 65.26, 110.50, 948.50, 2400.60, 5760.00, 1983.43, 2480.40, 250.45, 75.29, 3045.60], 

        'ord_date': ['2012-10-05', '2012-09-10', np.nan, '2012-08-17', '2012-09-10', '2012-07-27', '2012-09-10', '2012-10-10', '2012-10-10', '2012-06-27', '2012-04-25'], 

        'customer_id': [3002, 3001, 3001, 3003, 3002, 3001, 3001, 3004, 3003, 3002, 3001], 

        'salesman_id': [5002.0, 5003.0, 5001.0, np.nan, 5002.0, 5001.0, 5001.0, np.nan, 5003.0, 5002.0, 5003.0, np.nan]}

df = pd.DataFrame(data)

# Find and replace missing values with a placeholder (e.g., -1)

df_cleaned = df.fillna(-1)

print(df_cleaned)

```

```

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

import pandas as pd
import numpy as np

# Creating the DataFrame with missing values
data = {'ord_no': [70001.0, np.nan, 70002.0, 70004.0, np.nan, 70005.0, np.nan, 70010.0,
                  'purch_amt': [150.50, 270.65, 65.26, 110.50, 948.50, 2400.60, 5760.00, 1983.43,
                                2480.40, 250.45, 75.29, 3045.60],
                  'ord_date': ['2012-10-05', '2012-09-10', np.nan, '2012-08-17', '2012-09-10', '2012-07-27',
                               '2012-09-10', '2012-10-10', '2012-10-10', '2012-06-27', '2012-08-17', '2012-04-25'],
                  'customer_id': [3002, 3001, 3001, 3003, 3002, 3001, 3001, 3004, 3003, 3002, 3001],
                  'salesman_id': [5002.0, 5003.0, 5001.0, np.nan, 5002.0, 5001.0, 5001.0, 5003.0, 5002.0, 5003.0, np.nan]}

df = pd.DataFrame(data)

# Find and replace missing values with a placeholder (e.g., -1)
df_cleaned = df.fillna(-1)

# Display the cleaned DataFrame
print(df_cleaned)

```

```

RESTART: D:\Query processing\LAB QUESTION\LAB14.py
          ord_no  purch_amt  ord_date  customer_id  salesman_id
0      70001.0     150.50  2012-10-05        3002       5002.0
1          -1.0     270.65  2012-09-10        3001       5003.0
2      70002.0      65.26  2012-08-17        3001       5001.0
3      70004.0     110.50  2012-08-17        3003       5001.0
4          -1.0     948.50  2012-09-10        3002       5002.0
5      70005.0    2400.60  2012-07-27        3001       5001.0
6          -1.0     5760.00  2012-09-10        3001       5001.0
7      70010.0    1983.43  2012-10-10        3004       -1.0
8      70003.0    2480.40  2012-10-10        3003       5003.0
9      70012.0    250.45  2012-06-27        3002       5002.0
10         -1.0     75.29  2012-08-17        3001       5003.0
11     70013.0    3045.60  2012-04-25        3001       -1.0

```

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

CODE:

```

import pandas as pd

import numpy as np

data = {'ord_no': [70001.0, np.nan, 70002.0, 70004.0, np.nan, 70005.0, np.nan, 70010.0,
                  70003.0, 70012.0, np.nan, 70013.0],

                  'purch_amt': [150.50, 270.65, 65.26, 110.50, 948.50, 2400.60, 5760.00, 1983.43,
                                2480.40, 250.45, 75.29, 3045.60],

                  'ord_date': ['2012-10-05', '2012-09-10', np.nan, '2012-08-17', '2012-09-10', '2012-07-
                               27', '2012-09-10', '2012-10-10', '2012-10-10', '2012-06-27', '2012-08-17', '2012-04-25'],

                  'customer_id': [3002, 3001, 3001, 3003, 3002, 3001, 3001, 3004, 3003, 3002, 3001,
                                  3001],

                  'salesman_id': [5002.0, 5003.0, 5001.0, np.nan, 5002.0, 5001.0, 5001.0, 5003.0, 5002.0, 5003.0, np.nan]}

df = pd.DataFrame(data)

# Keep rows with at least 2 NaN values

df_filtered = df.dropna(thresh=2)

print(df_filtered)

```

```

LAB15.py - D:\Query processing\LAB QUESTION\LAB15.py (3.12.0)
File Edit Format Run Options Window Help
#Write a Pandas program to keep the rows with at least 2 NaN values in a given DataFrame.

import pandas as pd
import numpy as np

# Creating the DataFrame with missing values
data = {'ord_no': [70001.0, np.nan, 70002.0, 70004.0, np.nan, 70005.0, np.nan, 70010.0, 70003.0, 70012.0, np.nan, 70013.0],
        'purch_amt': [150.50, 270.65, 65.26, 110.50, 948.50, 2400.60, 570.00, 1983.43, 2480.40, 250.45, 75.29, 3045.60],
        'ord_date': ['2012-10-05', '2012-09-10', np.nan, '2012-08-17', '2012-07-27', '2012-07-27', '2012-09-10', '2012-10-10', '2012-10-10', '2012-06-27'],
        'customer_id': [3002, 3001, 3001, 3003, 3002, 3001, 3001, 3004, 3003, 3002, 3001, 3001],
        'salesman_id': [5002.0, 5003.0, 5001.0, np.nan, 5001.0, 5002.0, 5001.0, 5003.0, 5002.0, 5001.0, 5003.0, np.nan]}

df = pd.DataFrame(data)

# Keep rows with at least 2 NaN values
df_filtered = df.dropna(thresh=2)

# Display the filtered DataFrame
print(df_filtered)

```

IDLE Shell 3.12.0

```

File Edit Shell Debug Options Window Help
Python 3.12.0 (tags/v3.12.0:0fb18b0, Oct 2 2023, 13:03:39) [MSC v.1935 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.

>>> = RESTART: D:\query processing\LAB QUESTION\LAB15.py
      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

```

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

CODE:

```

import pandas as pd
from tabulate import tabulate
data = {
    'Student_ID': [1, 2, 3, 4, 5],
    'Name': ['Alice', 'Bob', 'Charlie', 'David', 'Emily'],
    'School_Code': ['S001', 'S002', 'S001', 'S002', 'S003'],
    'Grade': ['A', 'B', 'A', 'C', 'B']
}
df = pd.DataFrame(data)
grouped = df.groupby('School_Code')
all_groups = pd.concat([group for _, group in grouped])
print(tabulate(all_groups, headers='keys', tablefmt='pretty'))

```

```

LA816.py - D:\Query processing\LAB QUESTION\LAB16.py (3.12.0)
File Edit Format Run Options Window Help
#Write a Pandas program to split the following dataframe into groups based On school code. Also check the type of GroupBy object.

import pandas as pd
from tabulate import tabulate

data = {
    'Student_ID': [1, 2, 3, 4, 5],
    'Name': ['Alice', 'Bob', 'Charlie', 'David', 'Emily'],
    'School_Code': ['S001', 'S002', 'S001', 'S002', 'S003'],
    'Grade': ['A', 'B', 'A', 'C', 'B']
}

df = pd.DataFrame(data)
grouped = df.groupby('School_Code')
all_groups = pd.concat([group for _, group in grouped])
print(tabulate(all_groups, headers='keys', tablefmt='pretty'))

```

```

IDLE Shell 3.12.0
File Edit Shell Debug Options Window Help
Python 3.12.0 (tags/v3.12.0:0fb18b0, Oct 2 2023, 13:03:39) [MSC v.1935 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.

>>> = RESTART: D:\Query processing\LAB QUESTION\LAB16.py
+---+ | Student_ID | Name | School_Code | Grade |
+---+ | 0 | 1 | Alice | S001 | A |
| 2 | 3 | Charlie | S001 | A |
| 1 | 2 | Bob | S002 | B |
| 3 | 4 | David | S002 | C |
| 4 | 5 | Emily | S003 | B |
+---+
>>>

```

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.

CODE:

```

import pandas as pd

data = {

    'Student_ID': [1, 2, 3, 4, 5],
    'Name': ['Alice', 'Bob', 'Charlie', 'David', 'Emily'],
    'School_Code': ['S001', 'S002', 'S001', 'S002', 'S003'],
    'Age': [25, 30, 27, 28, 26]
}

df = pd.DataFrame(data)

school_stats = df.groupby('School_Code')['Age'].agg(['mean', 'min', 'max'])

print("Statistics for each school:")

print(school_stats)

```

```

import pandas as pd

data = {
    'Student_ID': [1, 2, 3, 4, 5],
    'Name': ['Alice', 'Bob', 'Charlie', 'David', 'Emily'],
    'School_Code': ['S001', 'S002', 'S001', 'S002', 'S003'],
    'Age': [23, 30, 27, 28, 26]
}

df = pd.DataFrame(data)
school_stats = df.groupby('School_Code')['Age'].agg(['mean', 'min', 'max'])

print("Statistics for each school:")
print(school_stats)

```

```

File Edit Shell Debug Options Window Help
Python 3.12.0 (tags/v3.12.0:0fb18b0, Oct 2 2023, 13:03:39) [MSC v.1935 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.

>>> = RESTART: D:\Query processing\LAB QUESTION\LAB17.py
Statistics for each school:
      mean  min  max
School_Code
S001      26.0   25   27
S002      29.0   28   30
S003      26.0   26   26
>>>

```

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

CODE:

```

import pandas as pd

data = {

    'Student_ID': [1, 2, 3, 4, 5, 6, 7, 8],
    'Name': ['Alice', 'Bob', 'Charlie', 'David', 'Emily', 'Frank', 'Grace', 'Henry'],
    'School_Code': ['S001', 'S002', 'S001', 'S002', 'S003', 'S001', 'S002', 'S003'],
    'Class': ['A', 'B', 'A', 'B', 'A', 'B', 'A', 'B']

}\

df = pd.DataFrame(data)

grouped = df.groupby(['School_Code', 'Class'])

for (school, class_), group in grouped:
    print(f"\nSchool: {school}, Class: {class_}")

    print(group)

```

```

import pandas as pd
data = {
    'Student_ID': [1, 2, 3, 4, 5, 6, 7, 8],
    'Name': ['Alice', 'Bob', 'Charlie', 'David', 'Emily', 'Frank', 'Grace', 'Henry'],
    'School_Code': ['S001', 'S002', 'S001', 'S002', 'S003', 'S001', 'S002', 'S003'],
    'Class': ['A', 'B', 'A', 'B', 'A', 'B', 'A', 'B']
}
df = pd.DataFrame(data)
grouped = df.groupby(['School_Code', 'Class'])
for (school, class_), group in grouped:
    print(f"\nSchool: {school}, Class: {class_}")
    print(group)

```

```

File Edit Shell Debug Options Window Help
[AMD64] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> = RESTART: D:\Query processing\LAB QUESTION\LAB18.py
School: S001, Class: A
  Student_ID   Name School_Code Class
0            1  Alice        S001     A
2            3 Charlie        S001     A

School: S001, Class: B
  Student_ID   Name School_Code Class
5            6 Frank        S001     B

School: S002, Class: A
  Student_ID   Name School_Code Class
6            7 Grace        S002     A
3            4 David        S002     B

School: S002, Class: B
  Student_ID   Name School_Code Class
1            2 Bob        S002     B
3            4 David        S002     B

School: S003, Class: A
  Student_ID   Name School_Code Class
4            5 Emily        S003     A

School: S003, Class: B
  Student_ID   Name School_Code Class
7            8 Henry        S003     B

```

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.

CODE:

```

import pandas as pd
from io import StringIO
csv_data = """
Year,WHO Region,Country,Type,Display Value
2015,Americas,United States,Beer,200
2015,Europe,France,Wine,300
2016,Africa,Nigeria,Spirits,150
2016,Americas,Canada,Beer,180
2017,Western Pacific,Japan,Spirits,250
"""
df = pd.read_csv(StringIO(csv_data))
# Display dimensions
print("Dimensions of the DataFrame:", df.shape)
# Extract column names
print("Column names:")
for col in df.columns:
    print(col)

```

```

LAB19.py - D:\Query processing\LAB QUESTION\LAB19.py (3.12.0)
File Edit Format Run Options Window Help
#Write a Pandas program to display the dimensions or shape of the World alcohol consumption dataset. Also extract the column names from the dataset.

import pandas as pd
from io import StringIO

csv_data = """
Year,WHO Region,Country,Type,Display Value
2015,Americas,United States,Beer,200
2015,Europe,France,Wine,300
2015,Africa,Nigeria,Spirits,150
2016,Americas,Canada,Beer,180
2017,Western Pacific,Japan,Spirits,250
"""
df = pd.read_csv(StringIO(csv_data))

# Display dimensions
print("Dimensions of the DataFrame:", df.shape)

# Extract column names
print("Column names:")
for col in df.columns:
    print(col)

```

IDLE Shell 3.12.0

```

File Edit Shell Debug Options Window Help
Python 3.12.0 (tags/v3.12.0:0fb18b0, Oct 2 2023, 13:03:39) [MSC v.1935 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.

>>>
= RESTART: D:\Query processing\LAB QUESTION\LAB19.py
Dimensions of the DataFrame: (5, 5)
Column names:
Year
WHO Region
Country
Type
Display Value
>>> |

```

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

CODE:

```

import pandas as pd

data = {'Column1': ['apple', 'banana', 'orange', 'grape', 'kiwi']}

df = pd.DataFrame(data)

substring_to_find = 'an'

matching_indices = df[df['Column1'].str.contains(substring_to_find)].index

print(f'Indices where '{substring_to_find}' appears in 'Column1': {matching_indices})

```

```

LAB20.py - D:\Query processing\LAB QUESTION\LAB20.py (3.12.0)
File Edit Format Run Options Window Help
#Write a Pandas program to find the index of a given substring of a DataFrame column.

import pandas as pd
data = {'Column1': ['apple', 'banana', 'orange', 'grape', 'kiwi']}
df = pd.DataFrame(data)
substring_to_find = 'an'
matching_indices = df[df['Column1'].str.contains(substring_to_find)].index
print(f'Indices where '{substring_to_find}' appears in 'Column1': {matching_indices}')

```

IDLE Shell 3.12.0

```

File Edit Shell Debug Options Window Help
Python 3.12.0 (tags/v3.12.0:0fb18b0, Oct 2 2023, 13:03:39) [MSC v.1935 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.

>>>
= RESTART: D:\Query processing\LAB QUESTION\LAB20.py
Indices where 'an' appears in 'Column1': Index([1, 2], dtype='int64')
>>> |

```

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

CODE:

```
import pandas as pd

data = {'Name': ['John', 'Alice', 'Bob', 'Emily'],
        'Age': [25, 30, 35, 40],
        'City': ['New York', 'Los Angeles', 'Chicago', 'San Francisco']}

df = pd.DataFrame(data)

print("Original DataFrame:")

print(df)

# Specify the column to swap cases

column_to_swap = 'Name'

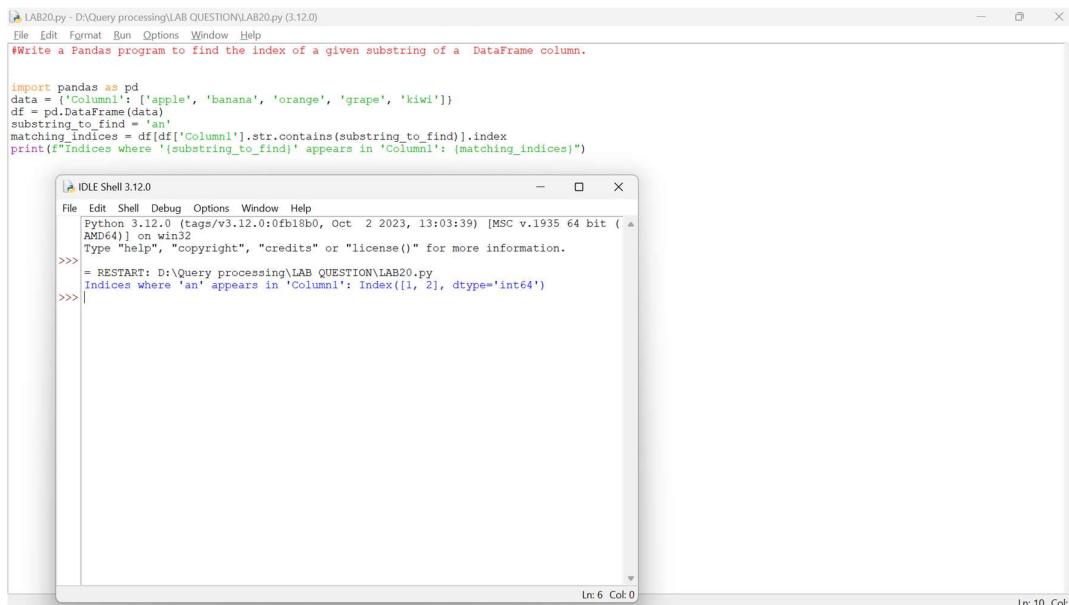
# Swap the case of the specified column

df[column_to_swap] = df[column_to_swap].apply(lambda x: x.swapcase() if
isinstance(x, str) else x)

# DataFrame after swapping cases

print("\nDataFrame after swapping cases:")

print(df)
```



The screenshot shows a Windows desktop environment with two windows open. The top window is titled 'LAB20.py - D:\Query processing\LAB QUESTION\LAB20.py (3.12.0)' and contains the Python code provided above. The bottom window is titled 'IDLE Shell 3.12.0' and shows the command-line interface where the script is run and its output is displayed. The output shows the original DataFrame, the modified DataFrame after swapping cases, and the results of a search operation.

```
LAB20.py - D:\Query processing\LAB QUESTION\LAB20.py (3.12.0)
File Edit Format Run Options Window Help
#Write a Pandas program to find the index of a given substring of a DataFrame column.

import pandas as pd
data = {'Column1': ['apple', 'banana', 'orange', 'grape', 'kiwi']}
df = pd.DataFrame(data)
substring_to_find = 'an'
matching_indices = df[df['Column1'].str.contains(substring_to_find)].index
print(f"Indices where '{substring_to_find}' appears in 'Column1': {matching_indices}")

IDLE Shell 3.12.0
File Edit Shell Debug Options Window Help
Python 3.12.0 (tags/v3.12.0:0fb18b0, Oct 2 2023, 13:03:39) [MSC v.1935 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> = RESTART: D:\Query processing\LAB QUESTION\LAB20.py
Indices where 'an' appears in 'Column1': Index([1, 2], dtype='int64')
>>>
```

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

CODE:

```
import matplotlib.pyplot as plt

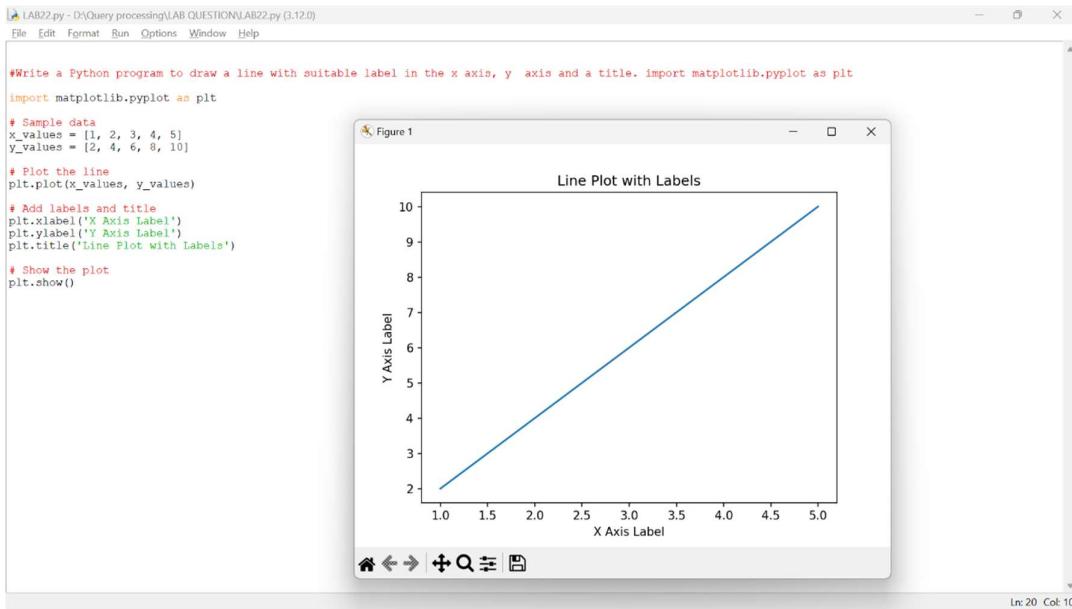
x_values = [1, 2, 3, 4, 5]
y_values = [2, 4, 6, 8, 10]

# Plot the line

plt.plot(x_values, y_values)

plt.xlabel('X Axis Label')
plt.ylabel('Y Axis Label')
plt.title('Line Plot with Labels')

plt.show()
```



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.

CODE:

```
import matplotlib.pyplot as plt

# Read data from the text file

data = []

with open('D:/Query processing/LAB QUESTION/test.txt', 'r') as file:

    for line in file:

        row = line.strip().split()
```

```

if len(row) == 2:
    data.append(row)

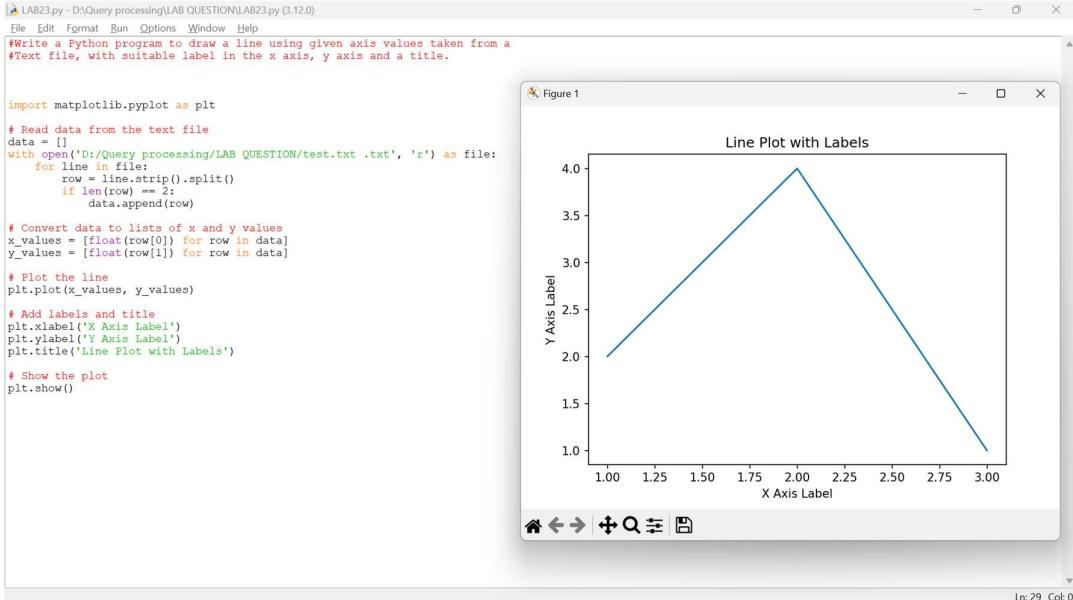
# Convert data to lists of x and y values
x_values = [float(row[0]) for row in data]
y_values = [float(row[1]) for row in data]

# Plot the line
plt.plot(x_values, y_values)

plt.xlabel('X Axis Label')
plt.ylabel('Y Axis Label')
plt.title('Line Plot with Labels')

plt.show()

```



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

CODE:

```

import pandas as pd

import matplotlib.pyplot as plt

from io import StringIO

csv_data = """Date,Open,High,Low,Close
10-03-16,774.25,776.065,002,769.5,772.559998
10-04-16,776.03,0029,778.71,0022,772.89,0015,776.429993

```

```
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"""
```

```
df = pd.read_csv(StringIO(csv_data), parse_dates=['Date'])

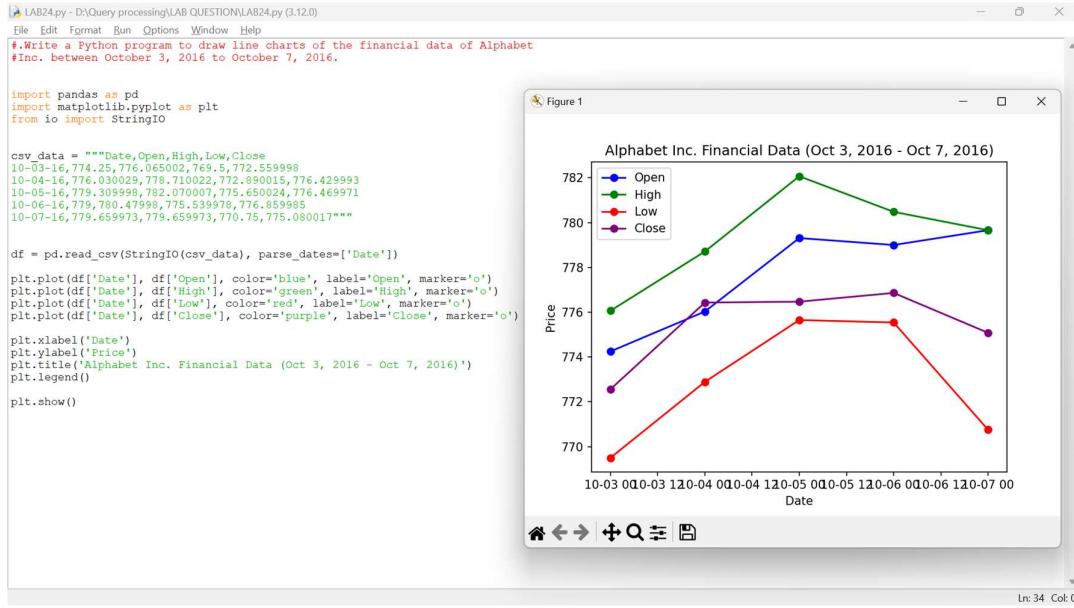
plt.plot(df['Date'], df['Open'], color='blue', label='Open', marker='o')
plt.plot(df['Date'], df['High'], color='green', label='High', marker='o')
plt.plot(df['Date'], df['Low'], color='red', label='Low', marker='o')
plt.plot(df['Date'], df['Close'], color='purple', label='Close', marker='o')

plt.xlabel('Date')
plt.ylabel('Price')

plt.title('Alphabet Inc. Financial Data (Oct 3, 2016 - Oct 7, 2016)')

plt.legend()

plt.show()
```



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

CODE:

```
import matplotlib.pyplot as plt

import numpy as np

x_values = np.linspace(0, 10, 100)

y_values_1 = np.sin(x_values)
```

```
y_values_2 = np.cos(x_values)
```

```
plt.plot(x_values, y_values_1, label='Line 1', color='blue', linewidth=3, linestyle='-', marker='D', markersize=8, markeredgecolor='blue', markerfacecolor='none')

plt.plot(x_values, y_values_2, label='Line 2', color='green', linewidth=5, linestyle='-', marker='D', markersize=8, markeredgecolor='green', markerfacecolor='none')

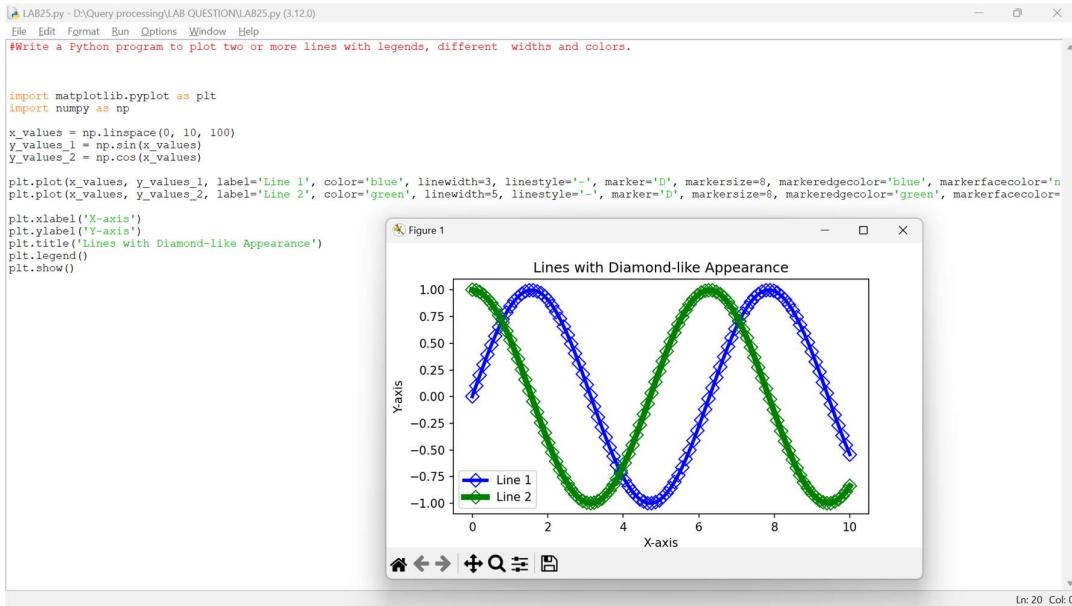
plt.xlabel('X-axis')

plt.ylabel('Y-axis')

plt.title('Lines with Diamond-like Appearance')

plt.legend()

plt.show()
```



26, Write a Python program to create multiple plots.

CODE:

```
import matplotlib.pyplot as plt

fig, axes = plt.subplots(nrows=2, ncols=2, figsize=(8, 8))

axes[0, 0].add_patch(plt.Rectangle((0.1, 0.1), 0.8, 0.8, fill=None, edgecolor='blue'))

axes[0, 0].set_title('Square 1')

axes[0, 1].add_patch(plt.Rectangle((0.1, 0.1), 0.8, 0.8, fill=None, edgecolor='green'))

axes[0, 1].set_title('Square 2')

axes[1, 0].add_patch(plt.Rectangle((0.1, 0.1), 0.8, 0.8, fill=None, edgecolor='red'))

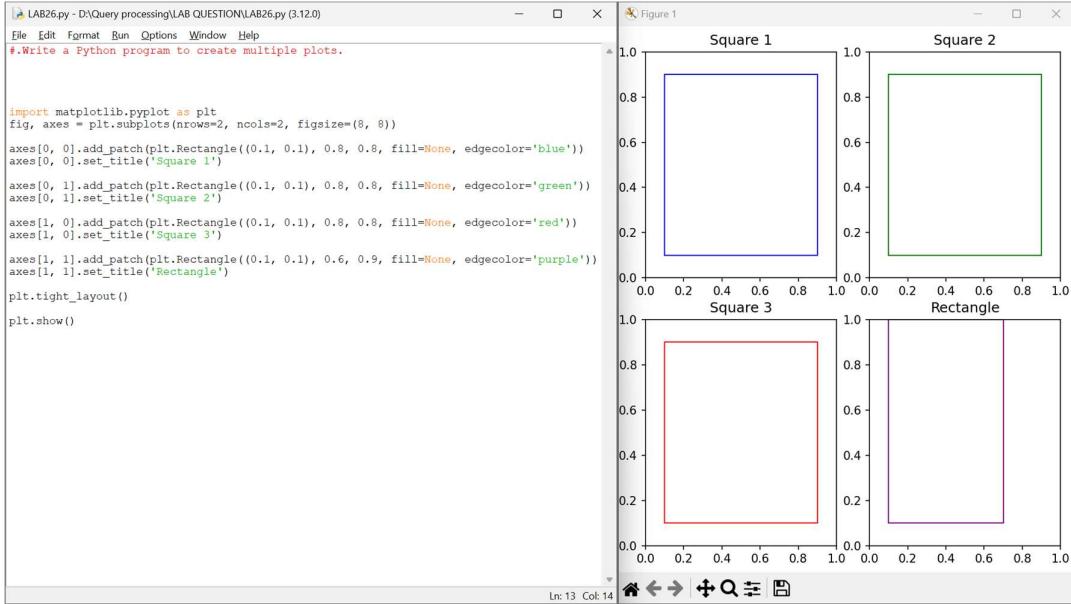
axes[1, 0].set_title('Square 3')

axes[1, 1].add_patch(plt.Rectangle((0.1, 0.1), 0.6, 0.9, fill=None, edgecolor='purple'))
```

```
axes[1, 1].set_title('Rectangle')
```

```
plt.tight_layout()
```

```
plt.show()
```



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

CODE:

```
import matplotlib.pyplot as plt

languages = ['Java', 'Python', 'PHP', 'JavaScript', 'C#', 'C++']

popularity = [22.2, 17.6, 8.8, 8, 7.7, 6.7]

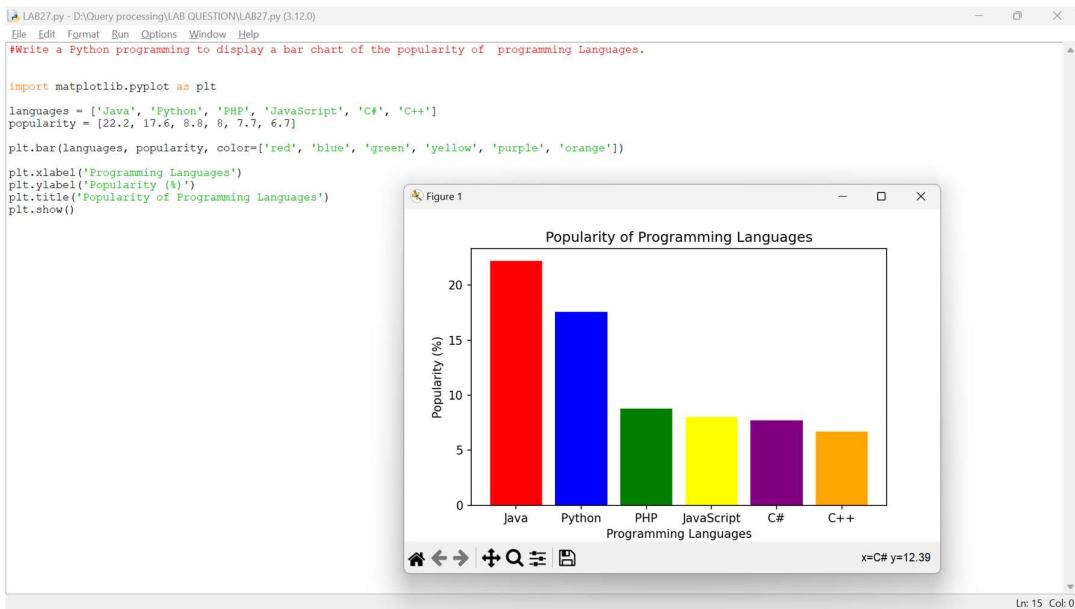
plt.bar(languages, popularity, color=['red', 'blue', 'green', 'yellow', 'purple', 'orange'])

plt.xlabel('Programming Languages')

plt.ylabel('Popularity (%)')

plt.title('Popularity of Programming Languages')

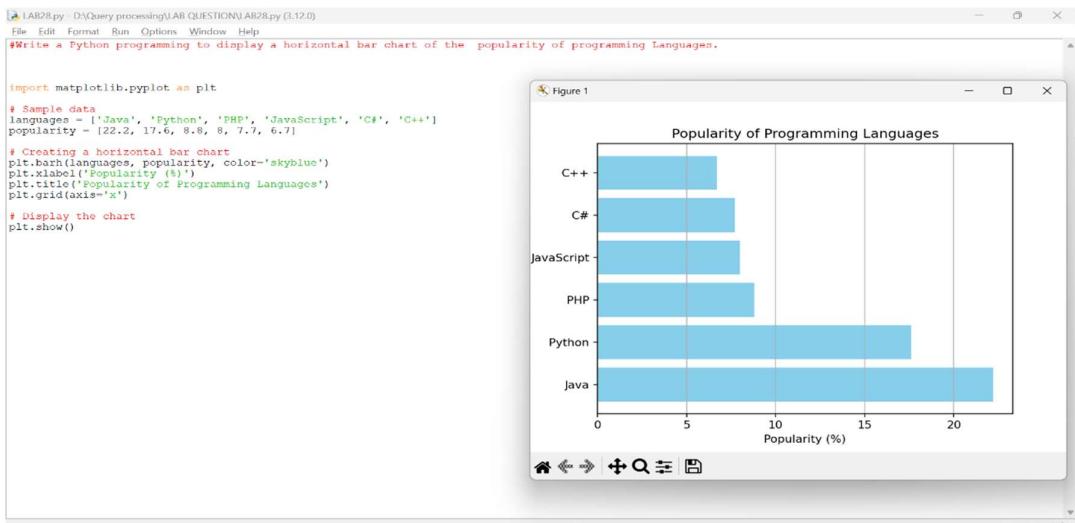
plt.show()
```



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

CODE:

```
import matplotlib.pyplot as plt
languages = ['Java', 'Python', 'PHP', 'JavaScript', 'C#', 'C++']
popularity = [22.2, 17.6, 8.8, 8, 7.7, 6.7]
# Creating a horizontal bar chart
plt.bart(languages, popularity, color='skyblue')
plt.xlabel('Popularity (%)')
plt.title('Popularity of Programming Languages')
plt.grid(axis='x')
plt.show()
```



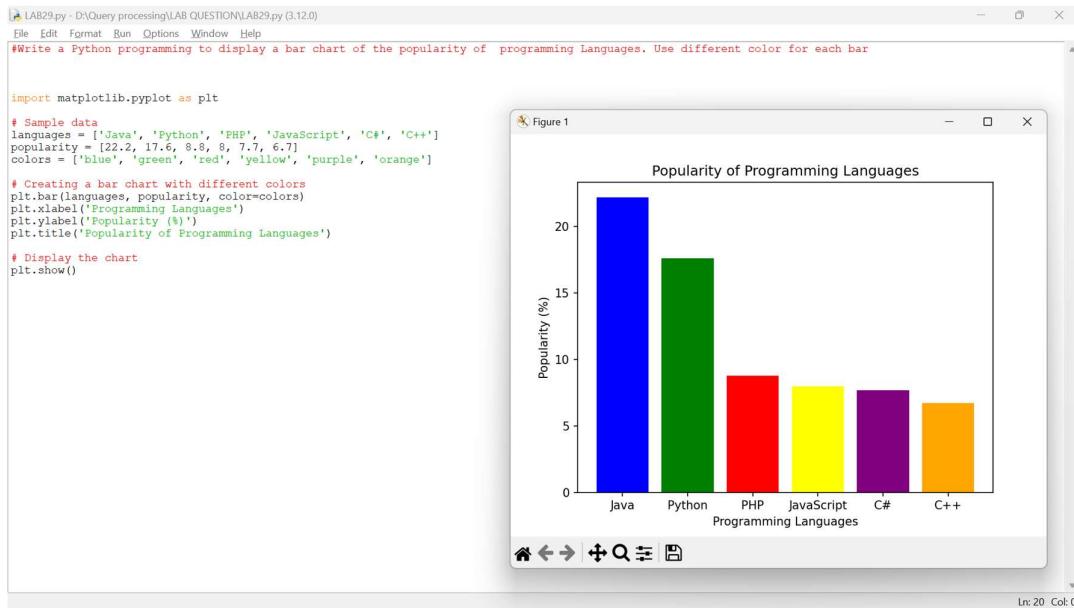
29, Write a Python programming to display a bar chart of the popularity of programming Languages. Use different color for each bar

CODE:

```
import matplotlib.pyplot as plt

languages = ['Java', 'Python', 'PHP', 'JavaScript', 'C#', 'C++']
popularity = [22.2, 17.6, 8.8, 8, 7.7, 6.7]
colors = ['blue', 'green', 'red', 'yellow', 'purple', 'orange']

# Creating a bar chart with different colors
plt.bar(languages, popularity, color=colors)
plt.xlabel('Programming Languages')
plt.ylabel('Popularity (%)')
plt.title('Popularity of Programming Languages')
plt.show()
```



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

CODE:

```
import matplotlib.pyplot as plt

import numpy as np

groups = ['Group 1', 'Group 2', 'Group 3', 'Group 4', 'Group 5']
means_men = [22, 30, 35, 35, 26]
means_women = [25, 32, 30, 35, 29]
```

```

x_values_men = np.arange(len(groups))

x_values_women = x_values_men + 0.4

plt.bar(x_values_men, means_men, width=0.4, label='Men', color='blue')

plt.bar(x_values_women, means_women, width=0.4, label='Women', color='pink')

plt.xlabel('Groups')

plt.ylabel('Scores')

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

plt.xticks(x_values_men + 0.2, groups)

plt.legend()

plt.show()

```

OUTPUT:



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

CODE:

```

import matplotlib.pyplot as plt

import numpy as np

groups = ['Group 1', 'Group 2', 'Group 3', 'Group 4', 'Group 5']

means_men = [22, 30, 35, 35, 26]

means_women = [25, 32, 30, 35, 29]

std_dev_men = [4, 3, 4, 1, 5]

std_dev_women = [3, 5, 2, 3, 3]

x_values = np.arange(len(groups))

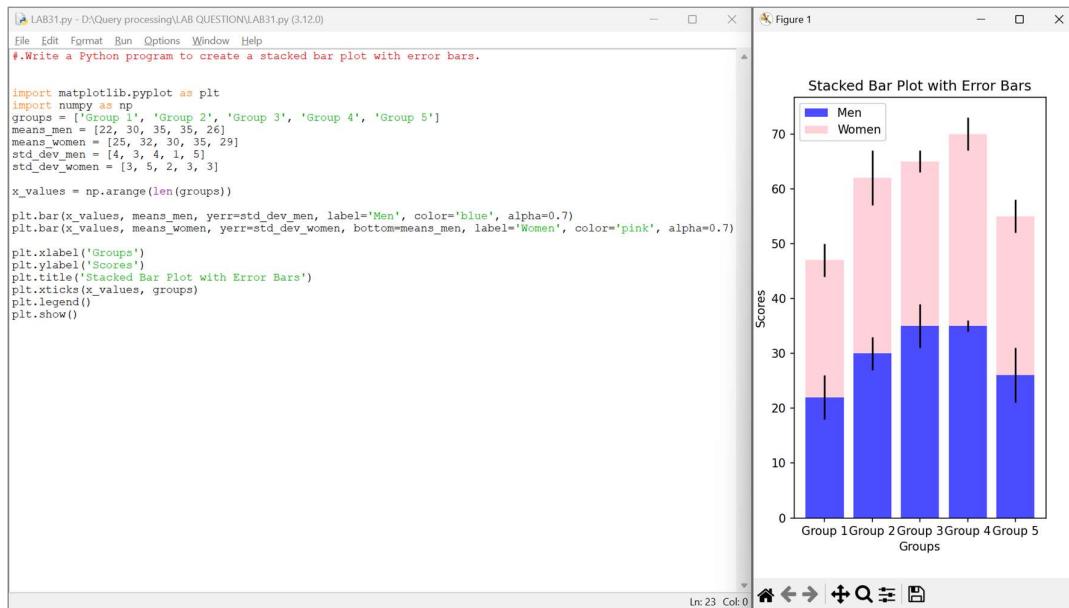
```

```

plt.bar(x_values, means_men, yerr=std_dev_men, label='Men', color='blue', alpha=0.7)
plt.bar(x_values, means_women, yerr=std_dev_women, bottom=means_men,
label='Women', color='pink', alpha=0.7)
plt.xlabel('Groups')
plt.ylabel('Scores')
plt.title('Stacked Bar Plot with Error Bars')
plt.xticks(x_values, groups)
plt.legend()
plt.show()

```

OUTPUT:



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

CODE:

```

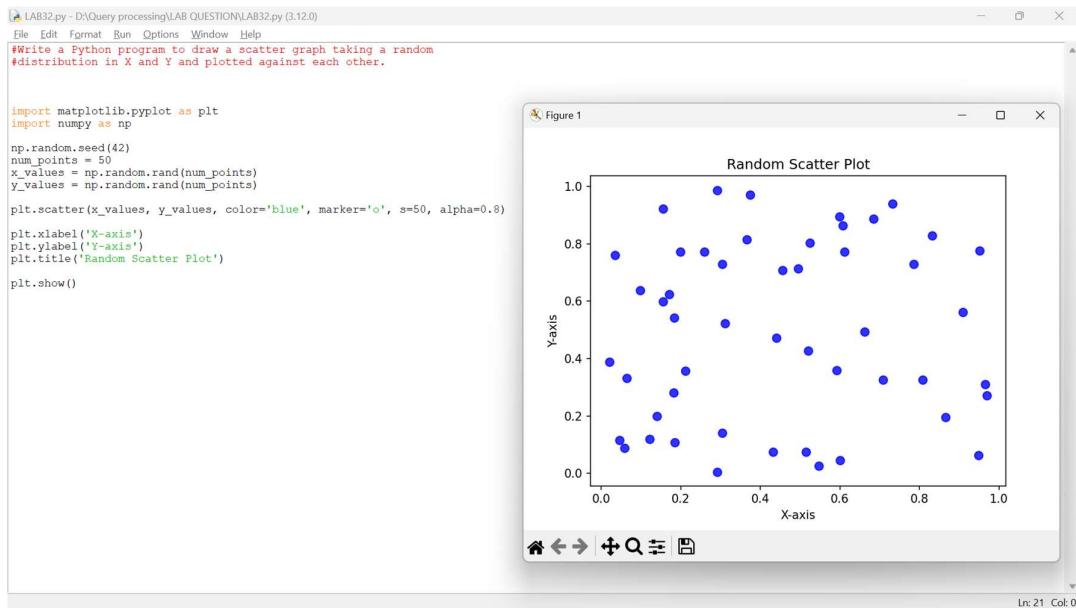
import matplotlib.pyplot as plt
import numpy as np
np.random.seed(42)
num_points = 50
x_values = np.random.rand(num_points)
y_values = np.random.rand(num_points)

```

```
plt.scatter(x_values, y_values, color='blue', marker='o', s=50, alpha=0.8)
```

```
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.title('Random Scatter Plot')
plt.show()
```

OUTPUT:

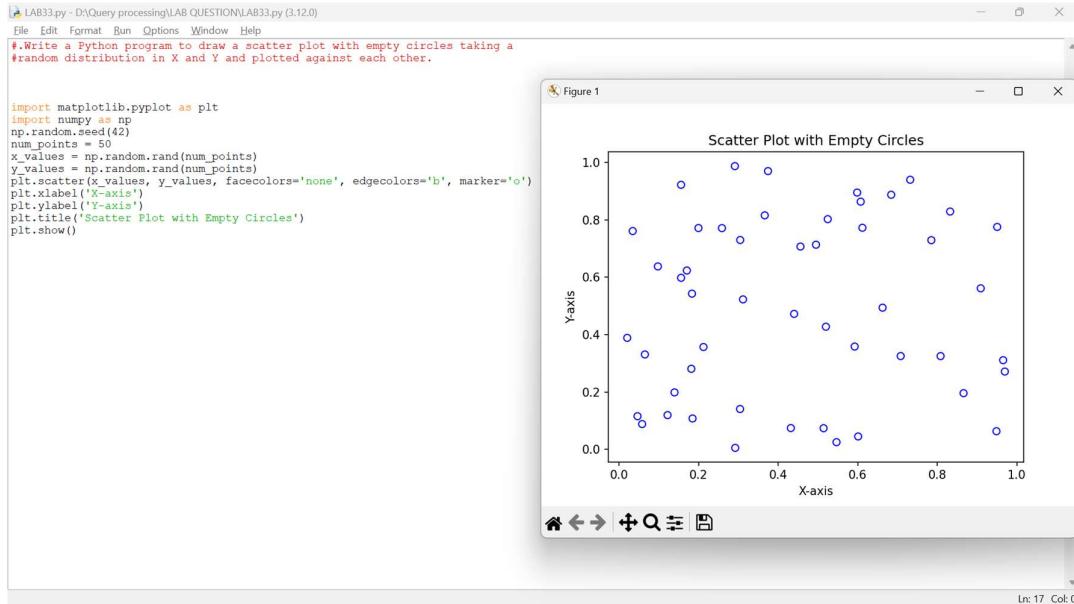


33, Write a Python program to draw a scatter plot with empty circles taking a random distribution in X and Y and plotted against each other.

CODE:

```
import matplotlib.pyplot as plt
import numpy as np
np.random.seed(42)
num_points = 50
x_values = np.random.rand(num_points)
y_values = np.random.rand(num_points)
plt.scatter(x_values, y_values, facecolors='none', edgecolors='b', marker='o')
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.title('Scatter Plot with Empty Circles')
plt.show()
```

OUTPUT:

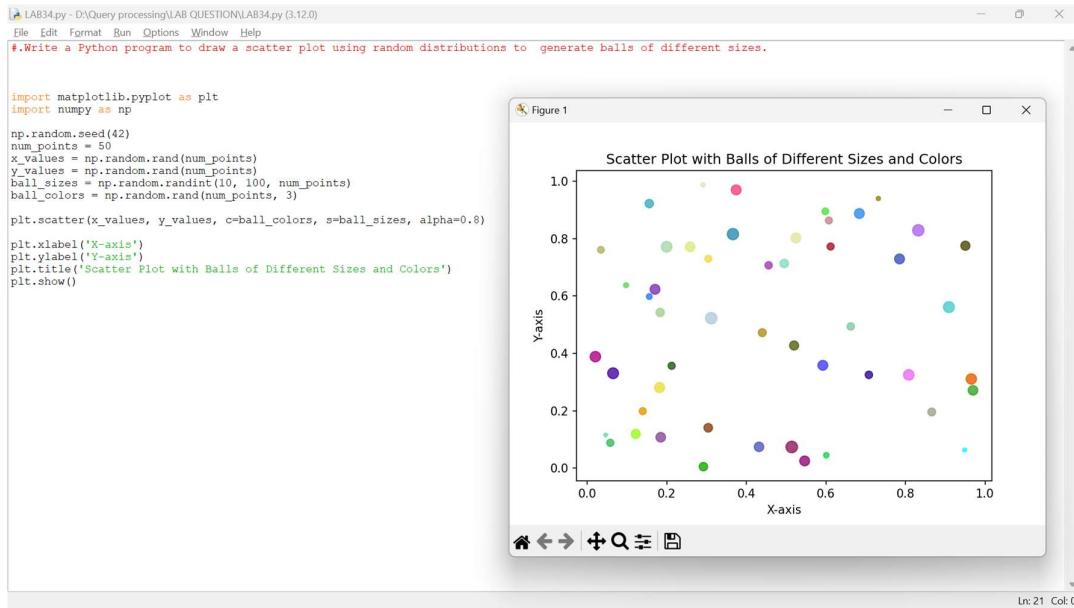


34. Write a Python program to draw a scatter plot using random distributions to generate balls of different sizes.

CODE:

```
import matplotlib.pyplot as plt  
import numpy as np  
np.random.seed(42)  
num_points = 50  
x_values = np.random.rand(num_points)  
y_values = np.random.rand(num_points)  
ball_sizes = np.random.randint(10, 100, num_points)  
ball_colors = np.random.rand(num_points, 3)  
plt.scatter(x_values, y_values, c=ball_colors, s=ball_sizes, alpha=0.8)  
plt.xlabel('X-axis')  
plt.ylabel('Y-axis')  
plt.title('Scatter Plot with Balls of Different Sizes and Colors')  
plt.show()
```

OUTPUT:



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

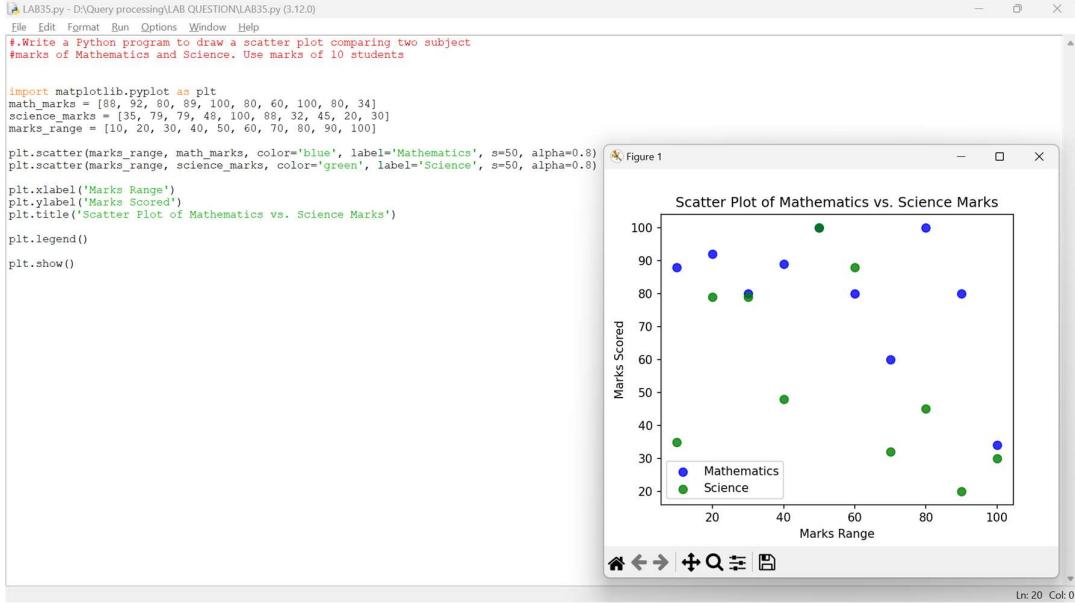
CODE:

```
import matplotlib.pyplot as plt

math_marks = [88, 92, 80, 89, 100, 80, 60, 100, 80, 34]
science_marks = [35, 79, 79, 48, 100, 88, 32, 45, 20, 30]
marks_range = [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]

plt.scatter(marks_range, math_marks, color='blue', label='Mathematics', s=50, alpha=0.8)
plt.scatter(marks_range, science_marks, color='green', label='Science', s=50, alpha=0.8)
plt.xlabel('Marks Range')
plt.ylabel('Marks Scored')
plt.title('Scatter Plot of Mathematics vs. Science Marks')
plt.legend()
plt.show()
```

OUTPUT:



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

CODE:

```
import matplotlib.pyplot as plt
import numpy as np

group1_heights = np.random.normal(170, 10, 30)
group1_weights = np.random.normal(65, 5, 30)

group2_heights = np.random.normal(160, 8, 30)
group2_weights = np.random.normal(55, 4, 30)

group3_heights = np.random.normal(175, 12, 30)
group3_weights = np.random.normal(75, 6, 30)

plt.scatter(group1_weights, group1_heights, color='Blue', label='Group 1', marker='*', s=50, alpha=0.8)

plt.scatter(group2_weights, group2_heights, color='Blue', label='Group 2', marker='*', s=50, alpha=0.8)

plt.scatter(group3_weights, group3_heights, color='Blue', label='Group 3', marker='*', s=50, alpha=0.8)

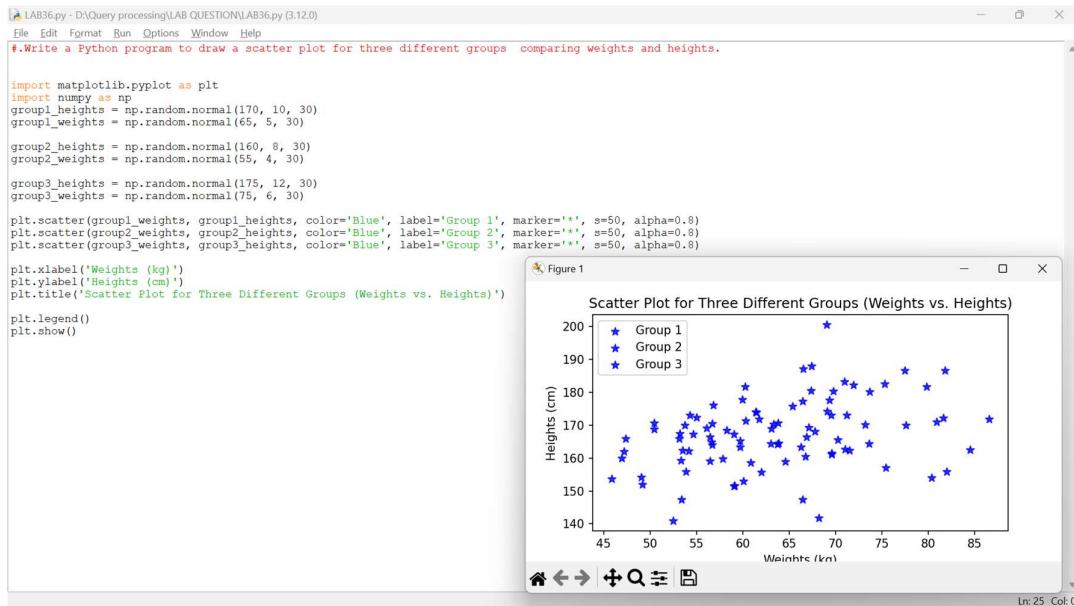
plt.xlabel('Weights (kg)')
plt.ylabel('Heights (cm)')
```

```
plt.title('Scatter Plot for Three Different Groups (Weights vs. Heights)')
```

```
plt.legend()
```

```
plt.show()
```

OUTPUT:



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

CODE:

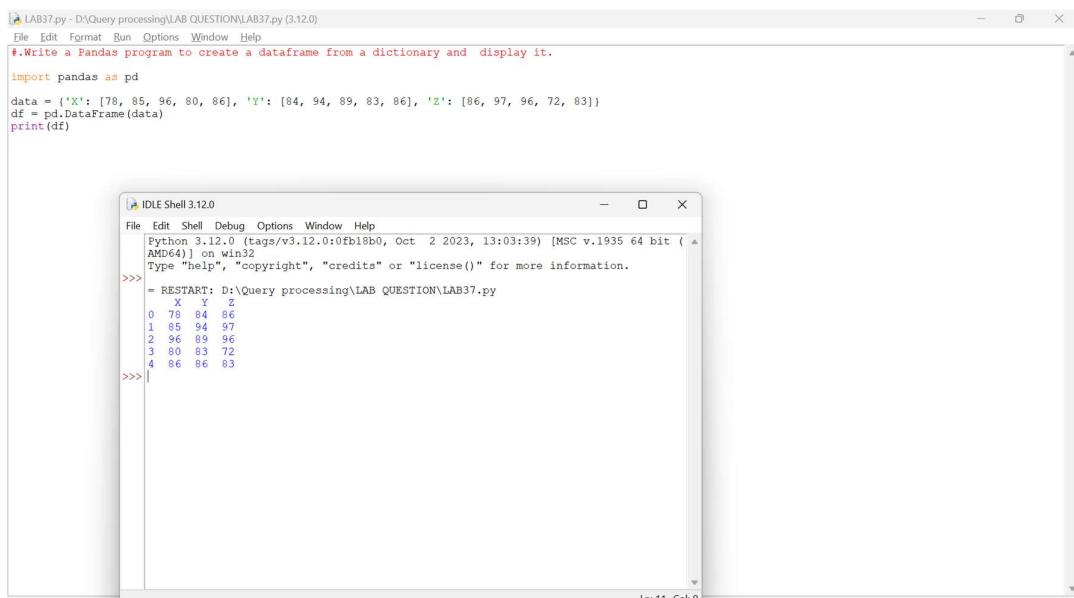
```
import pandas as pd
```

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

```
df = pd.DataFrame(data)
```

```
print(df)
```

OUTPUT:

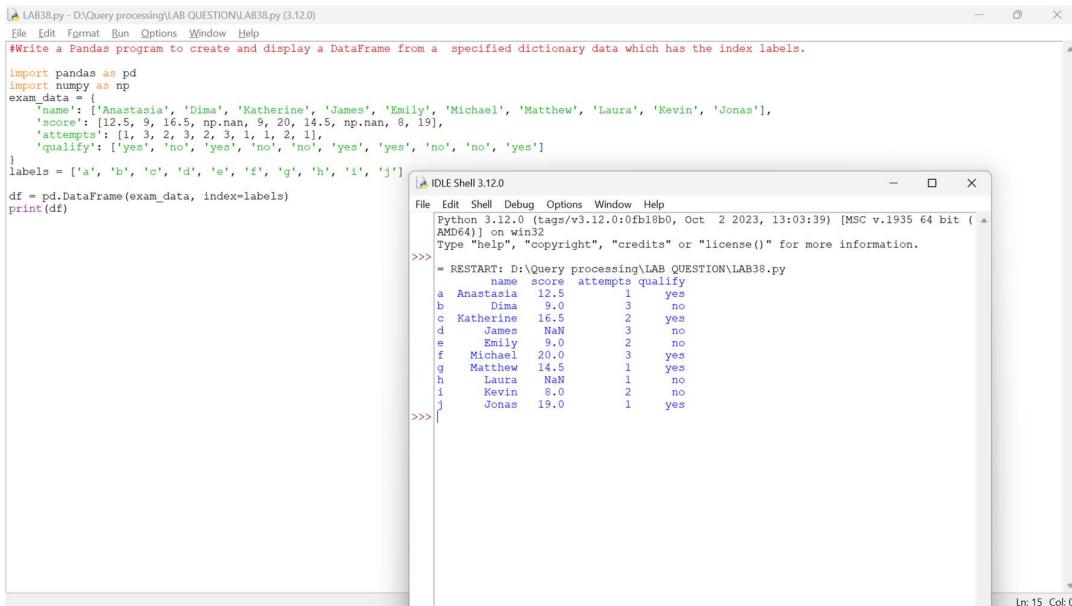


38. Write a Pandas program to create and display a DataFrame from a specified dictionary data which has the index labels.

CODE:

```
import pandas as pd
import numpy as np
exam_data = {
    'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael', 'Matthew', 'Laura', 'Kevin', 'Jonas'],
    'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19],
    'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],
    'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']
}
labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']
df = pd.DataFrame(exam_data, index=labels)
print(df)
```

OUTPUT:



The screenshot shows a Windows desktop environment. In the foreground, there is an 'IDLE Shell 3.12.0' window. The title bar says 'IDLE Shell 3.12.0'. The main area of the window displays the output of a Python script. The script defines a dictionary 'exam_data' with keys 'name', 'score', 'attempts', and 'qualify', and then creates a DataFrame 'df' using these. The resulting DataFrame is printed to the shell, showing the following data:

	name	score	attempts	qualify
a	Anastasia	12.5	3	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

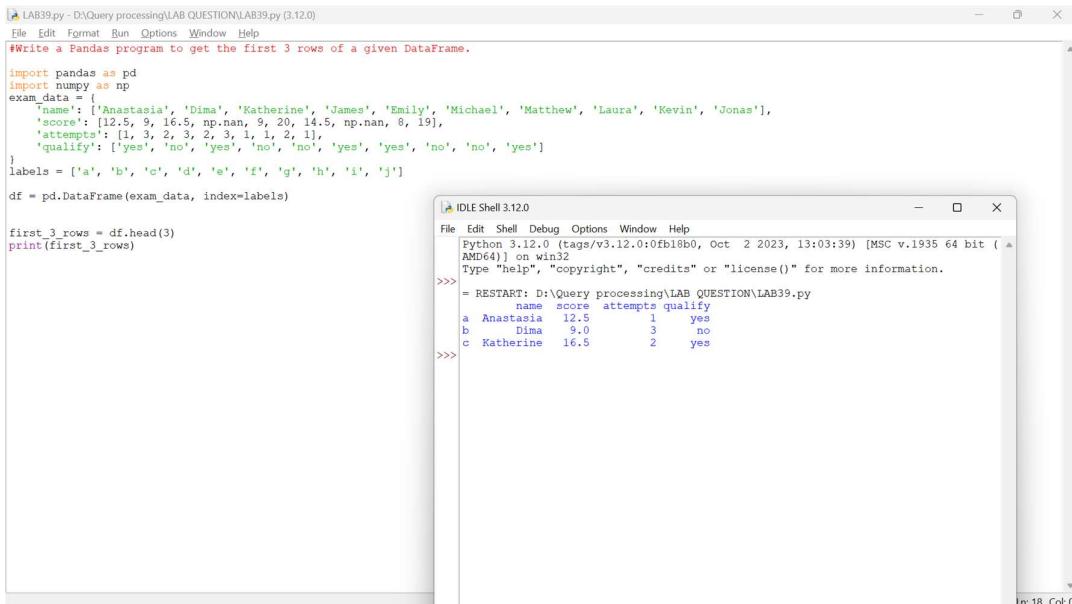
The shell also shows the command 'print(df)' and the prompt '>>>'. In the background, there is another window titled 'LAB38.py - D:\Query processing\LAB QUESTION\LAB38.py (3.12.0)'. This window contains the same Python code as the shell, with some syntax errors highlighted in red. The title bar of this window also includes the file path 'D:\Query processing\LAB QUESTION\LAB38.py (3.12.0)'.

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

CODE:

```
import pandas as pd
import numpy as np
exam_data = {
    'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael', 'Matthew', 'Laura',
'Kevin', 'Jonas'],
    'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19],
    'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],
    'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']
}
labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']
df = pd.DataFrame(exam_data, index=labels)
first_3_rows = df.head(3)
print(first_3_rows)
```

OUTPUT:



The screenshot shows a Windows desktop environment. In the foreground, there is an 'IDLE Shell 3.12.0' window. The title bar says 'IDLE Shell 3.12.0'. The menu bar includes File, Edit, Shell, Debug, Options, Window, and Help. Below the menu, it says 'Python 3.12.0 (tags/v3.12.0:0fb18b0, Oct 2 2023, 13:03:39) [MSC v.1935 64 bit (AMD64)] on win32'. It also says 'Type "help", "copyright", "credits" or "license()" for more information.' The main area of the window shows Python code and its output. The code is identical to the one in the question. The output shows the first three rows of the DataFrame:

```
= RESTART: D:\Query processing\LAB QUESTION\LAB39.py
   name      score  attempts  qualify
a Anastasia    12.5        1     yes
b Dima         9.0        3     no
c Katherine    16.5        2     yes
```

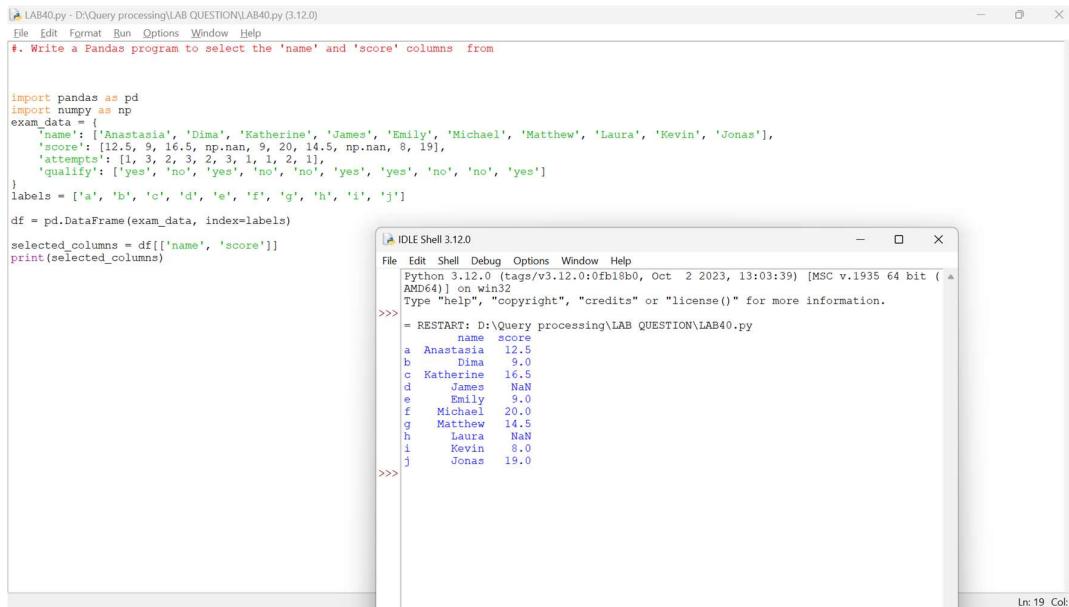
40. Write a Pandas program to select the 'name' and 'score' columns from

CODE:

```
import pandas as pd
import numpy as np
exam_data = {
    'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael', 'Matthew', 'Laura', 'Kevin', 'Jonas'],
    'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19],
    'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],
    'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']
}
labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']
df = pd.DataFrame(exam_data, index=labels)

selected_columns = df[['name', 'score']]
print(selected_columns)
```

OUTPUT:



The screenshot shows a Windows desktop environment. On the left, there is a code editor window titled "LAB40.py - D:\Query processing\LAB QUESTION\LAB40.py (3.12.0)". It contains the Python code provided above. On the right, there is an "IDLE Shell 3.12.0" window. This window shows the output of running the code. The output includes the imports, the definition of the exam_data dictionary, the creation of the df DataFrame, and the final print statement showing the selected columns.

```
import pandas as pd
import numpy as np
exam_data = {
    'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael', 'Matthew', 'Laura', 'Kevin', 'Jonas'],
    'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19],
    'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],
    'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']
}
labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']
df = pd.DataFrame(exam_data, index=labels)

selected_columns = df[['name', 'score']]
print(selected_columns)

RESTART: D:\Query processing\LAB QUESTION\LAB40.py
          name  score
a  Anastasia   12.5
b      Dima     9.0
c  Katherine   16.5
d      James    NaN
e      Emily    9.0
f      Michael  14.5
h      Laura    NaN
i      Kevin    8.0
j      Jonas   19.0
```