EXPERIMENT :1

AIM:-

Pandas program to select distinct department id from employes file.

CODE:-

1.py - C:/Users/Vishnu/Desktop/DSA0511-Query processing lab/1.py (3.11.0)

File Edit Format Run Options Window Help

```
import pandas as pd

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

df = pd.DataFrame(data)

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

print(distinct_department_ids)
```

INPUT:-

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

OUTPUT:-

```
IDLE Shell 3.11.0
File  Edit  Shell  Debug  Options  Window  Help
Python 3.11.0 (main, Oct 24 2022, 18:26:48) [MSC v.1933 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
====== RESTART: C:/Users/Vishnu/Desktop/DSA0511-Query processing lab/1.py ======
[ 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]
>>>
```

EXPERIMENT :2

AIM:-

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

CODE:-

2.py - C:/Users/Vishnu/Desktop/DSA0511-Query processing lab/2.py (3.11.0)

File Edit Format Run Options Window Help

```
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]
}

df = pd.DataFrame(data)

employee_counts = df['EMPLOYEE_ID'].value_counts()
employees_multiple_jobs = employee_counts[employee_counts > 1].index
print(employees_multiple_jobs)
```

INPUT:-

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

OUTPUT:-

IDLE Shell 3.11.0

File Edit Shell Debug Options Window Help

```
Python 3.11.0 (main, Oct 24 2022, 18:26:48) [MSC v.1933 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
====== RESTART: C:/Users/Vishnu/Desktop/DSA0511-Query processing lab/2.py ======
Index([101, 200, 176], dtype='int64', name='EMPLOYEE_ID')
>>>
```

EXPERIMENT :3

AIM:-

Pandas program to display the details of jobs in descending sequence on job title

CODE:-

3.py - C:/Users/Vishnu/Desktop/DSA0511-Query processing lab/3.py (3.11.0)

File Edit Format Run Options Window Help

```
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"
    "JOB_TITLE": ["President", "Administration Vice President", "Administration Assistant", "Finance Manager", "Accountant", "Accounting Manager", "Public Accountant", "Sales Manager", "Sal
    "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)

sorted_jobs = df.sort_values(by='JOB_TITLE', ascending=False)
print(sorted_jobs)
```

INPUT

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

OUTPUT

IDLE Shell 3.11.0

File Edit Shell Debug Options Window Help

```
Python 3.11.0 (main, Oct 24 2022, 18:26:48) [MSC v.1933 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
====== RESTART: C:/Users/Vishnu/Desktop/DSA0511-Query processing lab/3.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 Representative        4500       10500
6   AC_ACCOUNT                Public Accountant        4200        9000
14     IT_PROG                        Programmer        4000       10000
0      AD_PRES                         President       20080       40000
16      MK_REP          Marketing Representative        4000        9000
15      MK_MAN                 Marketing Manager        9000       15000
17      HR_REP   Human Resources Representative        4000        9000
3       FI_MGR                  Finance Manager        8200       16000
1        AD_VP    Administration Vice President       15000       30000
2      AD_ASST         Administration Assistant        3000        6000
5       AC_MGR                Accounting Manager        8200       16000
4   FI_ACCOUNT                        Accountant        4200        9000
>>>
```

EXPERIMENT :4

AIM

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

CODE

4.py - C:/Users/Vishnu/Desktop/DSA0511-Query processing lab/4.py (3.11.0)

File Edit Format Run Options Window Help

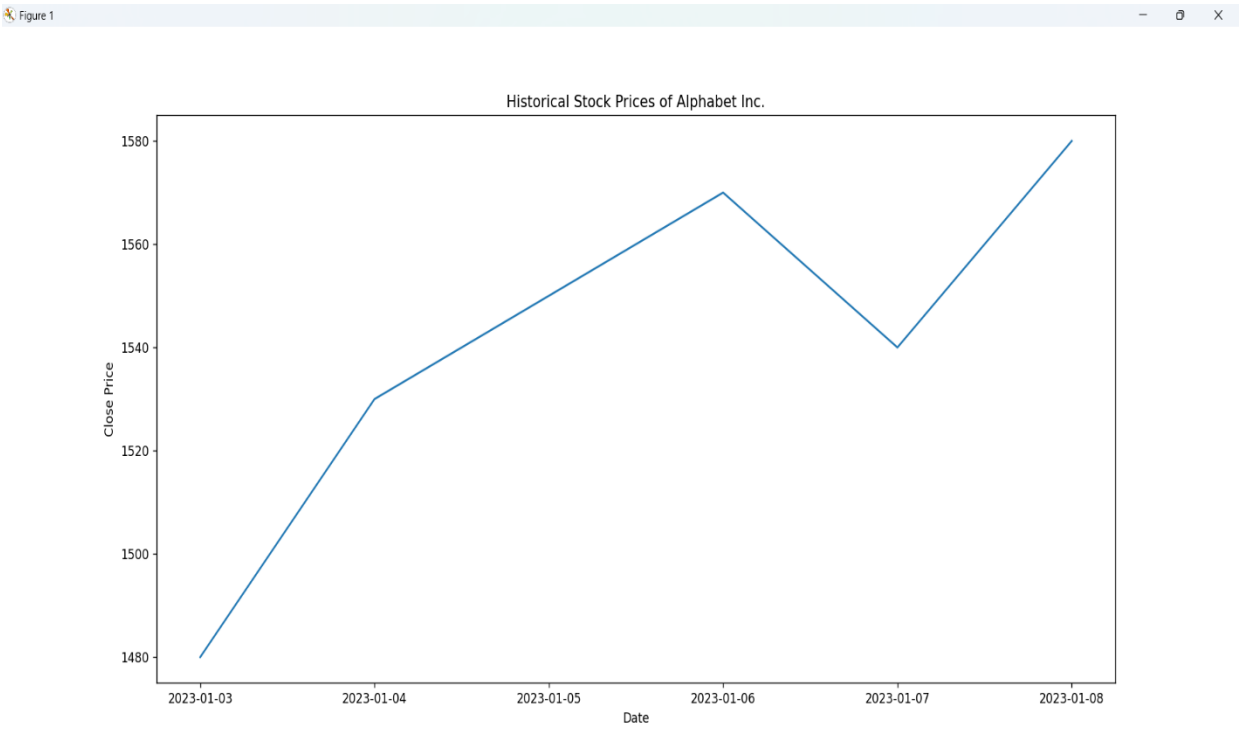
```
import pandas as pd
import matplotlib.pyplot as plt

data = {
    'Date': pd.date_range(start='2023-01-01', periods=10, freq='D'),
    'Close': [1500, 1520, 1480, 1530, 1550, 1570, 1540, 1580, 1600, 1620]
}

df = pd.DataFrame(data)
df['Date'] = pd.to_datetime(df['Date'])

start_date = '2023-01-03'
end_date = '2023-01-08'
mask = (df['Date'] >= start_date) & (df['Date'] <= end_date)
filtered_df = df.loc[mask]

plt.plot(filtered_df['Date'], filtered_df['Close'])
plt.xlabel('Date')
plt.ylabel('Close Price')
plt.title('Historical Stock Prices of Alphabet Inc.')
plt.show()
```

INPUT

Date = 2023-01-01

Close = [1500,1520,1480,1530,1550]

OUTPUT



EXPERIMENT :5

AIM

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

CODE

5.py - C:/Users/Vishnu/Desktop/DSA0511-Query processing lab/5.py (3.11.0)

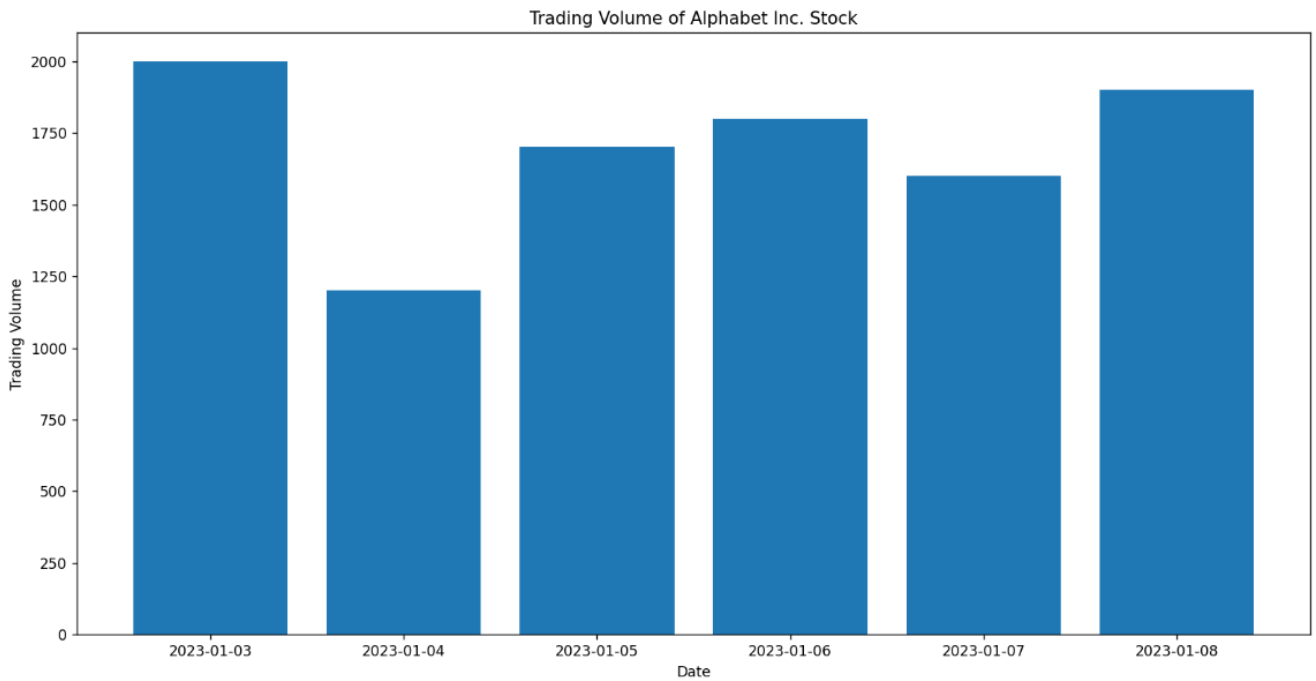
File Edit Format Run Options Window Help

```
import pandas as pd
import matplotlib.pyplot as plt

data = {
    'Date': pd.date_range(start='2023-01-01', periods=10, freq='D'),
    'Volume': [1000, 1500, 2000, 1200, 1700, 1800, 1600, 1900, 2100, 2200]
}

df = pd.DataFrame(data)
df['Date'] = pd.to_datetime(df['Date'])

start_date = '2023-01-03'
end_date = '2023-01-08'
mask = (df['Date'] >= start_date) & (df['Date'] <= end_date)
filtered_df = df.loc[mask]

plt.bar(filtered_df['Date'], filtered_df['Volume'])
plt.xlabel('Date')
plt.ylabel('Trading Volume')
plt.title('Trading Volume of Alphabet Inc. Stock')
plt.show()
```

INPUT

VOLUME:-[1000,1500,2000,1200,1700,1800,1600,1900]

OUTPUT



EXPERIMENT :6

AIM

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

CODE

6.py - C:/Users/Vishnu/Desktop/DSA0511-Query processing lab/6.py (3.11.0)

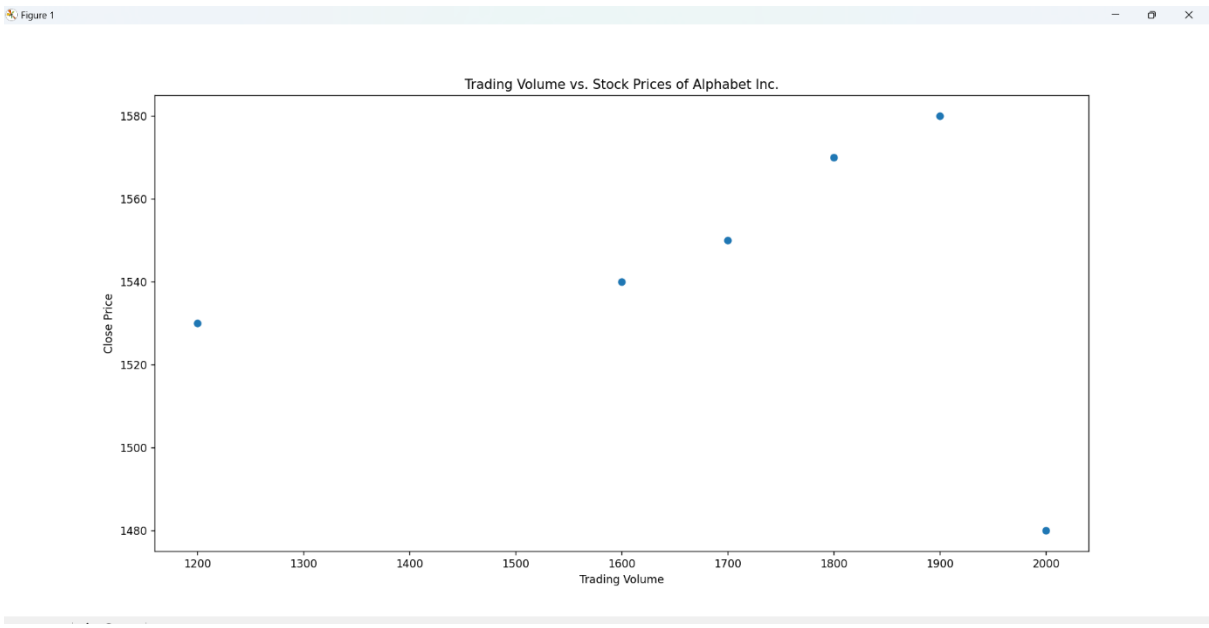
File Edit Format Run Options Window Help

```
import pandas as pd
import matplotlib.pyplot as plt

data = {
    'Date': pd.date_range(start='2023-01-01', periods=10, freq='D'),
    'Close': [1500, 1520, 1480, 1530, 1550, 1570, 1540, 1580, 1600, 1620],
    'Volume': [1000, 1500, 2000, 1200, 1700, 1800, 1600, 1900, 2100, 2200]
}

df = pd.DataFrame(data)
df['Date'] = pd.to_datetime(df['Date'])

start_date = '2023-01-03'
end_date = '2023-01-08'
mask = (df['Date'] >= start_date) & (df['Date'] <= end_date)
filtered_df = df.loc[mask]

plt.scatter(filtered_df['Volume'], filtered_df['Close'])
plt.xlabel('Trading Volume')
plt.ylabel('Close Price')
plt.title('Trading Volume vs. Stock Prices of Alphabet Inc.')
plt.show()
```

INPUT

Date	Open	High	Low	Close	Adj Close	Volume
01-04-2020	1122	1129.69	1097.45	1105.62	1105.62	2343100
02-04-2020	1098.26	1126.86	1096.4	1120.84	1120.84	1964900
03-04-2020	1119.015	1123.54	1079.81	1097.88	1097.88	2313400
06-04-2020	1138	1194.66	1130.94	1186.92	1186.92	2664700
07-04-2020	1221	1225	1182.23	1186.51	1186.51	2387300
08-04-2020	1206.5	1219.07	1188.16	1210.28	1210.28	1975100
09-04-2020	1224.08	1225.57	1196.735	1211.45	1211.45	2175400
13-04-2020	1209.18	1220.51	1187.598	1217.56	1217.56	1739800
14-04-2020	1245.09	1282.07	1236.93	1269.23	1269.23	2470400
15-04-2020	1245.61	1280.46	1240.4	1262.47	1262.47	1671700
16-04-2020	1274.1	1279	1242.62	1263.47	1263.47	2518100
17-04-2020	1284.85	1294.43	1271.23	1283.25	1283.25	1949000
20-04-2020	1271	1281.6	1261.37	1266.61	1266.61	1695500
21-04-2020	1247	1254.27	1209.71	1216.34	1216.34	2153000
22-04-2020	1245.54	1285.613	1242	1263.21	1263.21	2093100
23-04-2020	1271.55	1293.31	1265.67	1276.31	1276.31	1566200
24-04-2020	1261.17	1280.4	1249.45	1279.31	1279.31	1640400
27-04-2020	1296	1296.15	1269	1275.88	1275.88	1600600
28-04-2020	1287.93	1288.05	1232.2	1233.67	1233.67	2951300
29-04-2020	1341.46	1359.99	1325.34	1341.48	1341.48	3793600
30-04-2020	1324.88	1352.82	1322.49	1348.66	1348.66	2665400
01-05-2020	1328.5	1352.07	1311	1320.61	1320.61	2072500

OUTPUT



EXPERIMENT :7

AIM

Pandas program to create a Pivot table and find the maximum and minimum sale value of the items.(refer sales_data table)

CODE

7.py - C:/Users/Vishnu/Desktop/DSA0511-Query processing lab/7.py (3.11.0)

File Edit Format Run Options Window Help

```
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', 'West', 'Central', 'Central', 'Central', 'Central'],
    'Rep': ['Jones', 'Kivell', 'Jardine', 'Gill', 'Sorvino', 'Jones', 'Andrews', 'Jardine', 'Thompson', 'Jones',
            'Morgan', 'Howard', 'Parent', 'Smith', 'Gill', 'Andrews', 'Thompson', 'Jardine'],
    'Item': ['Pencil', 'Binder', 'Pencil', 'Pen', 'Pencil', 'Binder', 'Pencil', 'Binder', 'Pencil', 'Binder',
             'Binder', 'Binder', 'Binder', 'Desk', 'Pen Set', 'Binder', 'Pencil', 'Binder'],
    'Units': [95, 50, 36, 27, 56, 60, 75, 90, 32, 60, 55, 29, 81, 2, 42, 28, 32, 55],
    'UnitCost': [1.99, 19.99, 4.99, 19.99, 1.99, 4.99, 1.99, 19.99, 4.99, 4.99, 19.99, 4.99, 19.99, 125.00, 23.50, 4.99, 1.99, 19.99],
    'Total': [189.05, 999.50, 179.64, 539.73, 111.44, 299.40, 149.25, 1799.10, 159.68, 299.40, 1099.45, 144.71, 1619.19, 250.00, 993.00, 139.72, 63.68, 1099.45]
}

df = pd.DataFrame(data)

pivot_table = df.pivot_table(values='Total', index='Item', aggfunc=['max', 'min'])

print(pivot_table)
```

INPUT

Csv file (data frame)

OUTPUT

```
IDLE Shell 3.11.0
File  Edit  Shell  Debug  Options  Window  Help
Python 3.11.0 (main, Oct 24 2022, 18:26:48) [MSC v.1933 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
====== RESTART: C:/Users/Vishnu/Desktop/DSA0511-Query processing lab/7.py ======
             max     min
           Total   Total
Item
Binder   1799.10  139.72
Desk      250.00  250.00
Pen       539.73  539.73
Pen Set   993.00  993.00
Pencil    189.05   63.68
>>>
```

EXPERIMENT :8

AIM

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

CODE

8.py - C:/Users/Vishnu/Desktop/DSA0511-Query processing lab/8.py (3.11.0)

File Edit Format Run Options Window Help

```
import pandas as pd

data = {
    'Date': ['2023-06-01', '2023-06-01', '2023-06-02', '2023-06-02', '2023-06-03'],
    'Item': ['Apple', 'Banana', 'Apple', 'Banana', 'Apple'],
    'Units_Sold': [10, 5, 15, 8, 10],
    'Price_per_Unit': [0.5, 0.3, 0.5, 0.3, 0.5]
}

df = pd.DataFrame(data)

pivot_table = pd.pivot_table(df, values='Units_Sold', index='Item', aggfunc='sum')

print(pivot_table)
```

INPUT

Date = [2023-06-01,2023-06-01]

Item = [apple,banana,apple]

Sold = [10,15,11]

OUTPUT

```
IDLE Shell 3.11.0
File  Edit  Shell  Debug  Options  Window  Help
Python 3.11.0 (main, Oct 24 2022, 18:26:48) [MSC v.1933 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
====== RESTART: C:/Users/Vishnu/Desktop/DSA0511-Query processing lab/8.py ======
        Units_Sold
Item
Apple           35
Banana          13
>>>
```

EXPERIMENT :9

AIM

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

CODE

9.py - C:/Users/Vishnu/Desktop/DSA0511-Query processing lab/9.py (3.11.0)

File Edit Format Run Options Window Help

```
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', 'East', 'Central',
               'East', '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', '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]
}

df = pd.DataFrame(data)

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

print(pivot_table)
```

INPUT

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

OUTPUT

IDLE Shell 3.11.0

File Edit Shell Debug Options Window Help

```
Python 3.11.0 (main, Oct 24 2022, 18:26:48) [MSC v.1933 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
====== RESTART: C:/Users/Vishnu/Desktop/DSA0511-Query processing lab/9.py ======
                          Sale_amt
Region  Manager SalesMan
Central Douglas John         250.0
        Hermann Luis       150948.0
                Shelli      25000.0
                Sigal      121820.0
        Martha  Steven      89850.0
        Timothy David        6075.0
East    Douglas Karen       40500.0
        Martha  Alexander  231076.0
                Diana       14500.0
West    Douglas Michael     38336.0
        Timothy Stephen     67088.0
```

EXPERIMENT :10

AIM

Pandas program to highlight the negative numbers red and positive numbers black

CODE

10.py - C:\Users\ankal\Downloads\DSA0511-Query Processing\10.py (3.12.1)

File Edit Format Run Options Window Help

```
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)
```

INPUT

	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

OUTPUT

```
Python 3.12.1 (tags/v3.12.1:2305ca5, Dec  7 2023, 22:03:25) [MSC v.1937 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.

======= RESTART: C:\Users\ankal\Downloads\DSA0511-Query Processing\10.py =======
Original array:
     A         B         C         D         E
0  1.0  1.329212 -0.770033 -0.316280 -0.990810
1  2.0 -1.070816 -1.438713  0.564417  0.295722
2  3.0 -1.626404  0.219565  0.678805  1.889273
3  4.0  0.961538  0.104011 -0.481165  0.850229
4  5.0  1.453425  1.057737  0.165562  0.515018
5  6.0 -1.336936  0.562861  1.392855 -0.063328
6  7.0  0.121668  1.207603 -0.002040  1.627796
7  8.0  0.354493  1.037528 -0.385684  0.519818
8  9.0  1.686583 -1.325963  1.428984 -2.089354
9 10.0 -0.129820  0.631523 -0.586538  0.290720

Negative numbers red and positive numbers black:
```