

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

## INPUT:

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

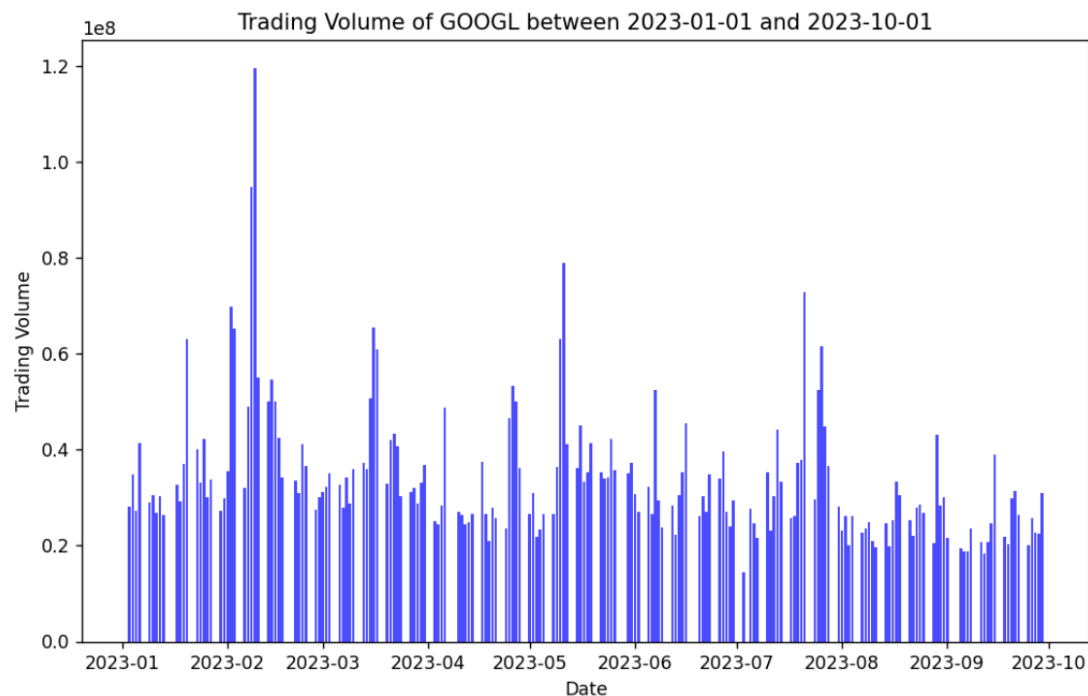
# Step 2: Retrieve historical stock data
ticker = "GOOGL" # Ticker symbol for Alphabet Inc.
start_date = "2023-01-01"
end_date = "2023-10-01"

# Using yfinance to get the stock data
data = yf.download(ticker, start=start_date, end=end_date)

# Step 3: Filter the data for the desired date range
# Since we're interested in trading volume, we only need that
column
data = data['Volume']

# Step 4: 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()
```

## OUTPUT:



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

## INPUT:

```
PROG9.PY - C:/Users/Supriya/OneDrive/Desktop/PROG9.PY (3.11.6)
File Edit Format Run Options Window Help
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',
    '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', 'Shellli', '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', 'Home Theater', '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, 500.00, 1198.00, 225.00, 1198.00, 500.00, 1198.00, 1198.00, 500.00,
    1198.00, 500.00, 500.00, 1198.00, 125.00, 58.50, 500.00, 225.00],
    'Sale_amt': [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:")
print(pivot_region)
print("\nTotal Sale Amount Manager-wise:")
print(pivot_manager)
print("\nTotal Sale Amount Salesman-wise:")
print(pivot_salesman)
```

## OUTPUT:

```
IDLE Shell 3.11.6
File Edit Shell Debug Options Window Help
Python 3.11.6 (tags/v3.11.6:8b6ee5b, Oct 2 2023, 14:57:12) [MSC v.1935 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:/Users/Supriya/OneDrive/Desktop/PROG9.PY =====
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 297768.0
Martha   235426.0
Timothy  73163.0

Total Sale Amount Salesman-wise:
      Sale_amt
SalesMan
Alexander 131076.0
David     6075.0
Diana    14500.0
John      250.0
Karen    40500.0
Luis     150948.0
Michael  38336.0
Shellli  25000.0
Sigal    121820.0
Stephen  67088.0
Steven   89850.0
>>>
```

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.

**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

**INPUT:**

```
PPROG10.PY - C:/Users/Supriya/OneDrive/Desktop/PPROG10.PY (3.11.6)
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)
```

## OUTPUT:

```
= RESTART: C:/Users/Supriya/OneDrive/Desktop/PPROG10.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:
```

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

## INPUT:

```
PROG11.PY - C:/Users/Supriya/OneDrive/Desktop/PROG11.PY (3.11.6)
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)
df.iloc[0, 2] = np.nan
df.iloc[3, 3] = np.nan
df.iloc[4, 1] = np.nan
df.iloc[9, 4] = np.nan
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.highlight_null(null_color='red')
```

## OUTPUT:

Original array:

	A	B	C	D	E
0	1.0	1.329212	NaN	-0.316280	-0.990810
1	2.0	-1.070816	-1.438713	0.564417	0.295722
2	3.0	-1.626404	0.219565	0.678805	1.889273
3	4.0	0.961538	0.104011	NaN	0.850229
4	5.0	NaN	1.057737	0.165562	0.515018
5	6.0	-1.336936	0.562861	1.392855	-0.063328
6	7.0	0.121668	1.207603	-0.002040	1.627796
7	8.0	0.354493	1.037528	-0.385684	0.519818
8	9.0	1.686583	-1.325963	1.428984	-2.089354
9	10.0	-0.129820	0.631523	-0.586538	NaN

Negative numbers red and positive numbers black:

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

INPUT:

```
PROG12.PY - C:/Users/Supriya/OneDrive/Desktop/PROG12.PY (3.11.6)
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)
df.iloc[0, 2] = np.nan
df.iloc[3, 3] = np.nan
df.iloc[4, 1] = np.nan
df.iloc[9, 4] = np.nan
print("Original array:")
print(df)
print("\nBackground: black - fontcolor: yellow")
df.style.set_properties(**{'background-color': 'black',
                          'color': 'yellow'})
```

Ln: 16 Col: 0

33°C Mostly cloudy 7:48 PM 6/13/2024

## OUTPUT:

Original array:

	A	B	C	D	E
0	1.0	1.329212	NaN	-0.316280	-0.990810
1	2.0	-1.070816	-1.438713	0.564417	0.295722
2	3.0	-1.626404	0.219565	0.678805	1.889273
3	4.0	0.961538	0.104011	NaN	0.850229
4	5.0	NaN	1.057737	0.165562	0.515018
5	6.0	-1.336936	0.562861	1.392855	-0.063328
6	7.0	0.121668	1.207603	-0.002040	1.627796
7	8.0	0.354493	1.037528	-0.385684	0.519818
8	9.0	1.686583	-1.325963	1.428984	-2.089354
9	10.0	-0.129820	0.631523	-0.586538	NaN

Background:black - fontcolor:yellow

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



## INPUT:

```
PROG13.PY - C:/Users/Supriya/OneDrive/Desktop/PROG13.PY (3.11.6)
File Edit Format Run Options Window Help
import pandas as pd
import numpy as np
pd.set_option('display.max_rows', None)
pd.set_option('display.max_columns', None)
df = pd.DataFrame({
    'ord_no': [70001, np.nan, 70002, 70004, np.nan, 70005, np.nan, 70010, 70003, 70012, np.nan, 70013],
    'purch_amt': [150.5, 270.65, 65.26, 110.5, 948.5, 2400.6, 5760, 1983.43, 2480.4, 250.45, 75.29, 3045.6],
    '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, 5003, 5001, np.nan, 5002, 5001, 5001, np.nan, 5003, 5002, 5003, np.nan]
})
print("Original Orders DataFrame:")
print(df)
print("\nMissing values of the said dataframe:")
print(df.isna())
```

## OUTPUT:

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

Missing values of the said dataframe:
   ord_no  purch_amt  ord_date  customer_id  salesman_id
0   False     False     False     False     False
1    True     False     False     False     False
2   False     False     True     False     False
3   False     False     False     False     True
4    True     False     False     False     False
5   False     False     False     False     False
6    True     False     False     False     False
7   False     False     False     False     True
8   False     False     False     False     False
9   False     False     False     False     False
10    True     False     False     False     False
11   False     False     False     False     True
```

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

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

**INPUT:**

```

prog14.py - C:/Users/Supriya/OneDrive/Desktop/prog14.py (3.11.6)
File Edit Format Run Options Window Help
import pandas as pd
import numpy as np
pd.set_option('display.max_rows', None)
#pd.set_option('display.max_columns', None)
df = pd.DataFrame({
'ord_no': [70001, np.nan, 70002, 70004, np.nan, 70005, "--", 70010, 70003, 70012, np.nan, 70013],
'purch_amt': [150.5, 270.65, 65.26, 110.5, 948.5, 2400.6, 5760, "?", 12.43, 2480.4, 250.45, 3045.6],
'ord_date': ['?', '2012-09-10', np.nan, '2012-08-17', '2012-09-10', '2012-07-27', '2012-09-10', '2012-10-10', '2012-10-10', '2012-06-27', '2012-08-17', '2012-04-25'],
'customer_id': [3002, 3001, 3001, 3003, 3002, 3001, 3001, 3004, "--", 3002, 3001, 3001],
'salesman_id': [5002, 5003, "?", 5001, np.nan, 5002, 5001, "?", 5003, 5002, 5003, "--"]}
print("Original Orders DataFrame:")
print(df)
print("\nReplace the missing values with NaN:")
result = df.replace({"?": np.nan, "--": np.nan})

```

## OUTPUT:

Original Orders DataFrame:

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

Replace the missing values with NaN:

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

## INPUT:

```
PROG15.PY - C:/Users/Supriya/OneDrive/Desktop/PROG15.PY (3.11.6)
File Edit Format Run Options Window Help
import pandas as pd
import numpy as np
pd.set_option('display.max_rows', None)
#pd.set_option('display.max_columns', None)
df = pd.DataFrame({
    'ord_no': [np.nan, np.nan, 70002, np.nan, np.nan, 70005, np.nan, 70010, 70003, 70012, np.nan, np.nan],
    'purch_amt': [np.nan, 270.65, 65.26, np.nan, 948.5, 2400.6, 5760, 1983.43, 2480.4, 250.45, 75.29, np.nan],
    'ord_date': [np.nan, '2012-09-10', np.nan, np.nan, '2012-09-10', '2012-07-27', '2012-09-10', '2012-10-10', '2012-10-10', '2012-06-27', '2012-08-17', np.nan],
    'customer_id': [np.nan, 3001, 3001, np.nan, 3002, 3001, 3001, 3004, 3003, 3002, 3001, np.nan]
})
print("Original Orders DataFrame:")
print(df)
print("\nKeep the rows with at least 2 NaN values of the said DataFrame:")
result = df.dropna(thresh=2)
print(result)
```

## OUTPUT:

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

Keep the rows with at least 2 NaN values of the said DataFrame:
   ord_no  purch_amt  ord_date  customer_id
1      NaN        270.65  2012-09-10        3001.0
2  70002.0         65.26         NaN        3001.0
4      NaN        948.50  2012-09-10        3002.0
5  70005.0       2400.60  2012-07-27        3001.0
6      NaN       5760.00  2012-09-10        3001.0
7  70010.0       1983.43  2012-10-10        3004.0
8  70003.0       2480.40  2012-10-10        3003.0
9  70012.0        250.45  2012-06-27        3002.0
10      NaN         75.29  2012-08-17        3001.0
> |
```

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

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

**INPUT:**

```
PROG16.PY - C:/Users/Supriya/OneDrive/Desktop/PROG16.PY (3.11.6)
File Edit Format Run Options Window Help
import pandas as pd
pd.set_option('display.max_rows', None)
#pd.set_option('display.max_columns', None)
student_data = pd.DataFrame({
    'school_code': ['s001', 's002', 's003', 's001', 's002', 's004'],
    'class': ['V', 'V', 'VI', 'VI', 'V', 'VI'],
    'name': ['Alberto Franco', 'Gino Mcneill', 'Ryan Parkes', 'Eesha Hinton', 'Gino Mcneill', 'David Parkes'],
    'date_Of_Birth': ['15/05/2002', '17/05/2002', '16/02/1999', '25/09/1998', '11/05/2002', '15/09/1997'],
    'age': [12, 12, 13, 13, 14, 12],
    'height': [173, 192, 186, 167, 151, 159],
    'weight': [35, 32, 33, 30, 31, 32],
    'address': ['street1', 'street2', 'street3', 'street1', 'street2', 'street4'],
    index=['S1', 'S2', 'S3', 'S4', 'S5', 'S6'])
print("Original DataFrame:")
print(student_data)
print('\nSplit the said data on school_code wise:')
result = student_data.groupby(['school_code'])
for name, group in result:
    print("\nGroup:")
    print(name)
```

## OUTPUT:

Original DataFrame:

	school_code	class	name	...	height	weight	address
S1	s001	V	Alberto Franco	...	173	35	street1
S2	s002	V	Gino Mcneill	...	192	32	street2
S3	s003	VI	Ryan Parkes	...	186	33	street3
S4	s001	VI	Eesha Hinton	...	167	30	street1
S5	s002	V	Gino Mcneill	...	151	31	street2
S6	s004	VI	David Parkes	...	159	32	street4

[6 rows x 8 columns]

Split the said data on school\_code wise:

Group:

('s001',)

Group:

('s002',)

Group:

('s003',)

Group:

('s004',)