ASSIGNMENT - 12(Pandas)

Solution/Ans by - Pranav Rode(29)

1. Write a Pandas program to replace all the NaN values with mean in a column of a DataFrame

In [1]:
```
import pandas as pd
import numpy as np

# Creating a sample DataFrame with NaN values
data = {'A': [1, 2, np.nan, 4, 5],
        'B': [np.nan, 2, 3, np.nan, 5]}

df = pd.DataFrame(data)

# Replace NaN values with mean in respective columns
df['A'].fillna(df['A'].mean(), inplace=True)
df['B'].fillna(df['B'].mean(), inplace=True)

df
```

Out[1]:

	A	B
0	1.0	3.333333
1	2.0	2.000000
2	3.0	3.000000
3	4.0	3.333333
4	5.0	5.000000

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

In [5]:
```
import pandas as pd

# Create a dictionary with data and index labels
data = {
    'Name': ['Alice', 'Bob', 'Charlie'],
    'Age': [25, 30, 35],
    'City': ['New York', 'San Francisco', 'Los Angeles']
}

# Specify the index labels
index_labels = ['Person1', 'Person2', 'Person3']

# Create a DataFrame from the dictionary with index labels
df = pd.DataFrame(data, index=index_labels)

# Display the DataFrame
df
```

Out[5]:

	Name	Age	City
Person1	Alice	25	New York
Person2	Bob	30	San Francisco
Person3	Charlie	35	Los Angeles

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

In [2]:
```
import pandas as pd

# Create a sample DataFrame
data = {'A': [1, 2, 3, 4, 5],
        'B': ['A', 'B', 'C', 'D', 'E']}

df = pd.DataFrame(data)

# Get the first 3 rows
df.head(3)
```

Out[2]:

	A	B
0	1	A
1	2	B
2	3	C

In [9]:
```
# Using indexing first 3 rows
df[:3]
```

Out[9]:

	A	B
0	1	A
1	2	B
2	3	C

4. Write a Pandas program to select the first 2 rows, 2 columns, and specific two columns

In [12]:
```
import pandas as pd

# Create a sample DataFrame
data = {'A': [1, 2, 3, 4, 5],
        'B': ['A', 'B', 'C', 'D', 'E'],
        'C': [10, 20, 30, 40, 50],
        'D': ['X', 'Y', 'Z', 'W', 'V']}

df = pd.DataFrame(data)

# Select the first 2 rows and specific 2 columns ('A' and 'C')
df.loc[:1, ['A', 'D']]
```

Out[12]:

	A	D
0	1	X
1	2	Y

5. Write a Pandas program to select the specified columns and rows from a given DataFrame

In [24]:
```
# Using loc
import pandas as pd

# Create a sample DataFrame
data = {'A': [1, 2, 3, 4, 5],
        'B': ['A', 'B', 'C', 'D', 'E'],
        'C': [10, 20, 30, 40, 50],
        'D': ['X', 'Y', 'Z', 'W', 'V']}

df = pd.DataFrame(data)

# Select specific rows (rows 1 and 3) and columns ('A' and 'C')
df.loc[[1, 3], ['A', 'C']]
```

Out[24]:

	A	C
1	2	20
3	4	40

In [25]:
```
# Using iloc
import pandas as pd

# Create a sample DataFrame
data = {'A': [1, 2, 3, 4, 5],
        'B': ['A', 'B', 'C', 'D', 'E'],
        'C': [10, 20, 30, 40, 50],
        'D': ['X', 'Y', 'Z', 'W', 'V']}

df = pd.DataFrame(data)

# Select specific rows (rows 1 and 3) and columns (columns 0 and 2)
df.iloc[[1, 3], [0, 2]]
```

Out[25]:

	A	C
1	2	20
3	4	40

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

In [33]:
```
import pandas as pd
import numpy as np

# Create a sample DataFrame
data = {'A': [1, np.nan, 3, 4, 5],
        'B': ['A', 'B', 'C', np.nan, 'E'],
        'C': [10, 20, 30, 40, 50],
        'D': ['X', 'Y', np.nan, 'W', np.nan]}
```

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

Out[33]:

	A	B	C	D
0	False	False	False	False
1	True	False	False	False
2	False	False	False	True
3	False	True	False	False
4	False	False	False	True

7. Write a Pandas program to count the number of missing values in each column of a given DataFrame

In [34]:
```
import pandas as pd
import numpy as np

# Create a sample DataFrame
data = {'A': [1, np.nan, 3, 4, 5],
        'B': ['A', 'B', 'C', np.nan, 'E'],
        'C': [10, 20, 30, 40, 50],
        'D': ['X', 'Y', np.nan, 'W', np.nan]}

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

Out[34]:
```
A    1
B    1
C    0
D    2
dtype: int64
```

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

- Example:
- Missing values: ?, --
- Replace those values with NaN

In [37]:
```
import pandas as pd
import numpy as np

# Create a sample DataFrame
data = {'A': [1, '?', 3, 4, 5],
        'B': ['A', 'B', '?', '--', 'E'],
        'C': [10, 20, 30, 40, 50],
        'D': ['X', '?', 'Z', 'W', '--']}

df = pd.DataFrame(data)

# Replace these values with NaN
df.replace(['?', '--'], np.nan, inplace=True)

# Display the DataFrame after replacing missing values
df
```

Out[37]:

	A	B	C	D
0	1.0	A	10	X
1	NaN	B	20	NaN
2	3.0	NaN	30	Z
3	4.0	NaN	40	W
4	5.0	E	50	NaN

9. Write a Pandas program to drop the rows where at least one element is missing in a given DataFrame

In [38]:
```
import pandas as pd

# Create a DataFrame
df = pd.DataFrame({'Name': ['John Doe', 'Jane Doe', np.nan, 'Peter Smith'],
                   'Age': [30, 25, 40, np.nan],
                   'Country': ['USA', 'Canada', np.nan, 'UK']})

# Drop rows where at least one element is missing
df.dropna()
```

Out[38]:

	Name	Age	Country
0	John Doe	30.0	USA
1	Jane Doe	25.0	Canada

10. Write a Pandas program to drop the rows where all elements are missing in a given DataFrame

In [51]:
```
import pandas as pd

# Create a DataFrame
df = pd.DataFrame({'Name': ['John Doe', 'Jane Doe', np.nan, np.nan, 'Peter Smith'], 'Age': [30, 25, np.nan, np.nan, 40], 'Country

# Drop rows where all elements are missing
df_dropped_rows = df.dropna(how='all')

# Print the DataFrame
df_dropped_rows
```

Out[51]:

	Name	Age	Country
0	John Doe	30.0	USA
1	Jane Doe	25.0	Canada
4	Peter Smith	40.0	UK

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

In [50]:
```
import pandas as pd
import numpy as np

# Creating a sample DataFrame
data = {
    'A': [1, np.nan, np.nan, 4, np.nan],
    'B': [np.nan, np.nan, 3, 4, 5],
    'C': [1, 2, 3, np.nan, np.nan]
}

df = pd.DataFrame(data)

# Display the original DataFrame
print("Original DataFrame:")
print(df)

# Keep rows with at least 2 NaN values
df[df.isnull().sum(axis=1) >= 2]
```

```
Original DataFrame:
     A    B    C
0  1.0  NaN  1.0
1  NaN  NaN  2.0
2  NaN  3.0  3.0
3  4.0  4.0  NaN
4  NaN  5.0  NaN
```

Out[50]:

	A	B	C
1	NaN	NaN	2.0
4	NaN	5.0	NaN

12. Write a Pandas program to keep the valid entries of a given DataFrame.

In [4]:
```
import pandas as pd
import numpy as np

# Create a sample DataFrame with some missing values
data = {'A': [1, 2, np.nan, 4, 5],
        'B': [np.nan, 2, 3, np.nan, 5],
        'C': [1, 2, 3, 4, 5]}

df = pd.DataFrame(data)

# Keep rows with valid (non-missing) entries
df_valid = df.dropna()

# Display the DataFrame with valid entries
print("DataFrame with valid entries:")
df_valid
```

```
DataFrame with valid entries:
```

Out[4]:

	A	B	C
1	2.0	2.0	2
4	5.0	5.0	5

13. Write a Pandas program to calculate the total number of missing values in a DataFrame

In [16]:
```
import pandas as pd
import numpy as np

# Create a sample DataFrame with some missing values
data = {'A': [1, 2, np.nan, 4, 5],
        'B': [np.nan, 2, 3, np.nan, 5],
        'C': [1, 2, 3, 4, 5]}

df = pd.DataFrame(data)

# Calculate the total number of missing values in each column
print('Of Each column:')
print(df.isna().sum() )
print('Total Missing values of whole dataframe =',df.isna().sum().sum())
```

```
Of Each column:
A    1
B    2
C    0
dtype: int64
Total Missing values of whole dataframe = 3
```

14. Write a Pandas program to replace NaNs with a single constant value in specified columns in a DataFrame

In [20]:
```
import pandas as pd
import numpy as np

# Create a sample DataFrame with some missing values
df = pd.DataFrame({'A': [1, 2, np.nan, 4, 5],
                   'B': [np.nan, 2, 3, np.nan, 5],
                   'C': [1, 2, 3, 4, 5]} )

# Replace NaNs with a constant value (e.g., 0) in column 'B'
df['B'].fillna(0, inplace=True)

# Display the DataFrame with NaNs replaced
print("DataFrame with NaNs replaced in column 'B':")
df
```

```
DataFrame with NaNs replaced in column 'B':
```

Out[20]:

	A	B	C
0	1.0	0.0	1
1	2.0	2.0	2
2	NaN	3.0	3
3	4.0	0.0	4
4	5.0	5.0	5

15. Write a Pandas program to replace NaNs with the median or mean of the specified columns in a given DataFrame.

In [23]:
```
import pandas as pd
import numpy as np

# Create a sample DataFrame with some missing values
df = pd.DataFrame({'A': [1, 2, np.nan, 4, 5],
                   'B': [np.nan, 2, 3, np.nan, 5],
                   'C': [1, 2, 3, 4, 5]})

# Replace NaNs with the median of column 'A'
median_A = df['A'].median()
df['A'].fillna(median_A, inplace=True)

# Replace NaNs with the mean of column 'B'
mean_B = df['B'].mean()
df['B'].fillna(mean_B, inplace=True)

# Display the DataFrame with NaNs replaced
print("DataFrame with NaNs replaced:")
df
```

```
DataFrame with NaNs replaced:
```

Out[23]:

	A	B	C
0	1.0	3.333333	1
1	2.0	2.000000	2
2	3.0	3.000000	3
3	4.0	3.333333	4
4	5.0	5.000000	5

16. Write a Pandas program to find the Indexes of missing values in a given DataFrame.

In [10]:
```
import pandas as pd
import numpy as np

# Create a sample DataFrame
df = pd.DataFrame({'A': [1, 2, np.nan, 4, 5],
                   'B': [np.nan, 2, 3, np.nan, 5],
                   'C': [1, 2, 3, 4, 5]} )

# Find the indexes of missing values
missing_indexes = df[df.isnull().any(axis=1)].index
print("Indexes of missing values:")
print(missing_indexes)
```

```
Indexes of missing values:
Int64Index([0, 2, 3], dtype='int64')
```

17. Write a Pandas program to select rows from a given DataFrame based on values in some columns.

In [9]:
```
import pandas as pd

# Create a sample DataFrame
df = pd.DataFrame({ 'Name': ['Alice', 'Bob', 'Charlie', 'David'],
                   'Age': [25, 30, 22, 35],
                   'Country': ['USA', 'Canada', 'USA', 'UK']} )

# Select rows where 'Country' is 'USA'
selected_rows = df[df['Country'] == 'USA']
print("Rows where 'Country' is 'USA':")
selected_rows
```

```
Rows where 'Country' is 'USA':
```

Out[9]:

	Name	Age	Country
0	Alice	25	USA
2	Charlie	22	USA

18. Write a Pandas program to change the order of a DataFrame columns.

In [14]:
```
import pandas as pd

# Create a sample DataFrame
df = pd.DataFrame({ 'Name': ['Alice', 'Bob', 'Charlie', 'David'],
                   'Age': [25, 30, 22, 35],
                   'Country': ['USA', 'Canada', 'USA', 'UK']} )
# Change the order of columns
df1 = df[["Country", "Name", "Age"]]
df1
```

Out[14]:

	Country	Name	Age
0	USA	Alice	25
1	Canada	Bob	30
2	USA	Charlie	22
3	UK	David	35

19. Write a Pandas program to add one row in an existing DataFrame

In [40]:
```
import pandas as pd

# Create a sample DataFrame
df = pd.DataFrame({ 'Name': ['Alice', 'Bob', 'Charlie', 'David'],
                   'Age': [25, 30, 22, 35],
                   'Country': ['USA', 'Canada', 'USA', 'UK']} )
df = df.append({'Name':'Sam', 'Age':28, 'Country':'Mexico'}, ignore_index=True)
df
```

```
C:\Users\prana\AppData\Local\Temp\ipykernel_8984\2981242863.py:7: FutureWarning: The frame.append method is deprecated and will
be removed from pandas in a future version. Use pandas.concat instead.
  df = df.append({'Name':'Sam', 'Age':28, 'Country':'Mexico'}, ignore_index=True)
```

Out[40]:

	Name	Age	Country
0	Alice	25	USA
1	Bob	30	Canada
2	Charlie	22	USA
3	David	35	UK
4	Sam	28	Mexico

20. Write a Pandas program to delete DataFrame row(s) based on a given column value

In [41]:
```
import pandas as pd

# Create a sample DataFrame
df = pd.DataFrame({ 'Name': ['Alice', 'Bob', 'Charlie', 'David'],
                    'Age': [25, 30, 22, 35]})

# Delete rows where 'Age' is 30
df = df[df['Age'] != 30]
print("DataFrame with rows where 'Age' is not 30:")
df
```

```
DataFrame with rows where 'Age' is not 30:
```

Out[41]:

	Name	Age
0	Alice	25
2	Charlie	22
3	David	35

21. Write a Pandas program to select a row of series/DataFrame by given integer index

In [57]:
```
import pandas as pd

# Create a sample DataFrame
df = pd.DataFrame({ 'Name': ['Alice', 'Bob', 'Charlie', 'David'],
                    'Age': [25, 30, 22, 35],
                    'Country': ['USA', 'Canada', 'USA', 'UK']} )
df.iloc[[1,3]]      # Getting rows of only index numbered 1 and 3
```

Out[57]:

	Name	Age	Country
1	Bob	30	Canada
3	David	35	UK

22. Write a Pandas program to get the length of the string present in a given column in a DataFrame

In [59]:
```
import pandas as pd

# Create a sample DataFrame
df = pd.DataFrame({'Name': ['Alice', 'Bob', 'Charlie', 'David'],
                   'City': ['New York', 'Los Angeles', 'Chicago', 'Houston']})

# Get the length of strings in the 'City' column
df['City_Length'] = df['City'].str.len()
print("DataFrame with string lengths:")
df
```

```
DataFrame with string lengths:
```

Out[59]:

	Name	City	City_Length
0	Alice	New York	8
1	Bob	Los Angeles	11
2	Charlie	Chicago	7
3	David	Houston	7

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

In [60]:
```
import pandas as pd

# Create a DataFrame
df = pd.DataFrame({'Name': ['John Doe', 'Jane Doe', 'Peter Smith']})
```

```
# Swap the cases of the characters in the 'Name' column
df['Name'] = df['Name'].str.swapcase()
df
```

Out[60]:

	Name
0	jOHN dOE
1	jANE dOE
2	pETER sMITH

24. Write a Pandas program to convert a specified character column in upper/lower cases in a given DataFrame.

In [63]:
```
import pandas as pd

# Create a DataFrame
df = pd.DataFrame({'Name': ['John Doe', 'Jane Doe', 'Peter Smith']})

# Convert the 'Name' column to upper case
df['Name'] = df['Name'].str.upper()

print(df)

# Convert the 'Name' column to lower case
df['Name'] = df['Name'].str.lower()

print(df)
```

```
          Name
0     JOHN DOE
1     JANE DOE
2  PETER SMITH
          Name
0     john doe
1     jane doe
2  peter smith
```

25. Write a Pandas program to remove whitespaces, left-sided whitespaces, and right-sided whitespaces of the string values of a given pandas series

In [82]:
```
import pandas as pd

# Create a sample Pandas Series
data = {'Text': ['    John  Doe  ', '  Jane Doe', ' Peter Smith   ']}
series = pd.Series(data['Text'])

# Remove leading whitespaces
series_stripped = series.str.strip()

# Remove all whitespaces from string values
series_stripped1 = series_stripped.str.replace(' ', '')

print("Original Series:")
print(series)
print("\nSeries with All Leading and Trailing Whitespaces Removed:")
print(series_stripped)
print("\nSeries with All Whitespaces Removed:")
print(series_stripped1)
```

```
Original Series:
0        John  Doe
1           Jane Doe
2     Peter Smith
dtype: object

Series with All Leading and Trailing Whitespaces Removed:
0      John  Doe
1       Jane Doe
2    Peter Smith
dtype: object

Series with All Whitespaces Removed:
0       JohnDoe
1       JaneDoe
2    PeterSmith
dtype: object
```

26. Write a Pandas program to extract years between 1800 to 2200 from the specified column of a given DataFrame.

df = pd.DataFrame({

'company_code': ['c0001','c0002','c0003', 'c0003', 'c0004'],

'year': ['year 1800','year 1700','year 2300', 'year 1900', 'year 2200']

})

In [96]:
```
import pandas as pd

# Create the DataFrame
df = pd.DataFrame({ 'company_code': ['c0001', 'c0002', 'c0003', 'c0003', 'c0004'],
                    'year': ['year 1800', 'year 1700', 'year 2300', 'year 1900', 'year 2200']
                  })

# Use regular expression to extract years between 1800 and 2200
df['year'] = df['year'].str.extract(r'(\b\d{4}\b)')  # Extract 4-digit numbers

# Convert the extracted year column to integers
df['year'] = df['year'].astype(int)

# Filter for years between 1800 and 2200
filtered_df = df[(df['year'] >= 1800) & (df['year'] <= 2200)]

print("Filtered DataFrame:")
filtered_df
```

Out[96]:

Filtered DataFrame:

	company_code	year
0	c0001	1800
3	c0003	1900
4	c0004	2200

27. Write a Pandas program to join the two given dataframes along rows

In [8]:
```
import pandas as pd

student_data1 = pd.DataFrame({
        'student_id': ['S1', 'S2', 'S3', 'S4', 'S5'],
         'name': ['Danniella Fenton', 'Ryder Storey', 'Bryce Jensen', 'Ed Bernal', 'Kwame Morin'],
        'marks': [200, 210, 190, 222, 199]})

student_data2 = pd.DataFrame({
        'student_id': ['S4', 'S5', 'S6', 'S7', 'S8'],
        'name': ['Scarlette Fisher', 'Carla Williamson', 'Dante Morse', 'Kaiser William', 'Madeeha Preston'],
        'marks': [201, 200, 198, 219, 201]})

print("Original DataFrames:")
print(student_data1)
print("-------------------------------------")
print(student_data2)
print("\nJoin the said two dataframes along rows:")
result_data = pd.concat([student_data1, student_data2])
print(result_data)
```

```
Original DataFrames:
  student_id              name  marks
0         S1  Danniella Fenton    200
1         S2      Ryder Storey    210
2         S3      Bryce Jensen    190
3         S4         Ed Bernal    222
4         S5       Kwame Morin    199
-------------------------------------
  student_id              name  marks
0         S4  Scarlette Fisher    201
1         S5  Carla Williamson    200
2         S6       Dante Morse    198
3         S7    Kaiser William    219
4         S8   Madeeha Preston    201

Join the said two dataframes along rows:
  student_id              name  marks
0         S1  Danniella Fenton    200
1         S2      Ryder Storey    210
2         S3      Bryce Jensen    190
3         S4         Ed Bernal    222
4         S5       Kwame Morin    199
0         S4  Scarlette Fisher    201
1         S5  Carla Williamson    200
2         S6       Dante Morse    198
3         S7    Kaiser William    219
4         S8   Madeeha Preston    201
```

In [10]:
```
import pandas as pd

student_data1 = pd.DataFrame({
        'student_id': ['S1', 'S2', 'S3', 'S4', 'S5'],
         'name': ['Danniella Fenton', 'Ryder Storey', 'Bryce Jensen', 'Ed Bernal', 'Kwame Morin'],
        'marks': [200, 210, 190, 222, 199]})

student_data2 = pd.DataFrame({
        'student_id': ['S4', 'S5', 'S6', 'S7', 'S8'],
        'name': ['Scarlette Fisher', 'Carla Williamson', 'Dante Morse', 'Kaiser William', 'Madeeha Preston'],
        'marks': [201, 200, 198, 219, 201]})

print("Original DataFrames:")
```

```
print(student_data1)
print("-----------------------------------")
print(student_data2)
print("\nJoining two dataframes along rows:")
result_data = pd.concat([student_data1, student_data2])
result_data
```

```
Original DataFrames:
  student_id              name  marks
0         S1  Danniella Fenton    200
1         S2      Ryder Storey    210
2         S3      Bryce Jensen    190
3         S4         Ed Bernal    222
4         S5       Kwame Morin    199
-----------------------------------
  student_id              name  marks
0         S4  Scarlette Fisher    201
1         S5  Carla Williamson    200
2         S6       Dante Morse    198
3         S7    Kaiser William    219
4         S8   Madeeha Preston    201

Joining two dataframes along rows:
```

Out[10]:

	student_id	name	marks
0	S1	Danniella Fenton	200
1	S2	Ryder Storey	210
2	S3	Bryce Jensen	190
3	S4	Ed Bernal	222
4	S5	Kwame Morin	199
0	S4	Scarlette Fisher	201
1	S5	Carla Williamson	200
2	S6	Dante Morse	198
3	S7	Kaiser William	219
4	S8	Madeeha Preston	201

28. Write a Pandas program to join the two given dataframes along with columns

In [13]:
```
import pandas as pd

student_data1 = pd.DataFrame({
        'student_id': ['S1', 'S2', 'S3', 'S4', 'S5'],
         'name': ['Danniella Fenton', 'Ryder Storey', 'Bryce Jensen', 'Ed Bernal', 'Kwame Morin'],
        'marks': [200, 210, 190, 222, 199]})

student_data2 = pd.DataFrame({
        'student_id': ['S4', 'S5', 'S6', 'S7', 'S8'],
        'name': ['Scarlette Fisher', 'Carla Williamson', 'Dante Morse', 'Kaiser William', 'Madeeha Preston'],
        'marks': [201, 200, 198, 219, 201]})

print("Original DataFrames:")
print(student_data1)
print("-----------------------------------")
print(student_data2)
print("\nJoining two dataframes along rows:")
result_data = pd.concat([student_data1, student_data2], axis=1)
result_data
```

```
Original DataFrames:
  student_id              name  marks
0         S1  Danniella Fenton    200
1         S2      Ryder Storey    210
2         S3      Bryce Jensen    190
3         S4         Ed Bernal    222
4         S5       Kwame Morin    199
-----------------------------------
  student_id              name  marks
0         S4  Scarlette Fisher    201
1         S5  Carla Williamson    200
2         S6       Dante Morse    198
3         S7    Kaiser William    219
4         S8   Madeeha Preston    201

Joining two dataframes along rows:
```

Out[13]:

	student_id	name	marks	student_id	name	marks
0	S1	Danniella Fenton	200	S4	Scarlette Fisher	201
1	S2	Ryder Storey	210	S5	Carla Williamson	200
2	S3	Bryce Jensen	190	S6	Dante Morse	198
3	S4	Ed Bernal	222	S7	Kaiser William	219
4	S5	Kwame Morin	199	S8	Madeeha Preston	201

29. Write a Pandas program to join the two given dataframes along rows and merge with another dataframe along with the common column id.

In [18]:
```
import pandas as pd

# Create the first DataFrame
df1 = pd.DataFrame({'id': [1, 2, 3],
                    'A': ['A1', 'A2', 'A3'],
                    'B': ['B1', 'B2', 'B3']})

# Create the second DataFrame
df2 = pd.DataFrame({'id': [4, 5, 6],
                    'A': ['A4', 'A5', 'A6'],
                    'B': ['B4', 'B5', 'B6']})

# Join the first two DataFrames along rows
result_df = pd.concat([df1, df2])
print(result_df)

# Create the third DataFrame
df3 = pd.DataFrame({'id': [2, 4, 6],
                    'C': ['C2', 'C4', 'C6']})

# Merge the result_df with df3 based on the 'id' column
merged_df = pd.merge(result_df, df3, on='id')

print("Resulting DataFrame after joining and merging:")
merged_df
```

```
   id   A   B
0   1  A1  B1
1   2  A2  B2
2   3  A3  B3
0   4  A4  B4
1   5  A5  B5
2   6  A6  B6
Resulting DataFrame after joining and merging:
```

Out[18]:

	id	A	B	C
0	2	A2	B2	C2
1	4	A4	B4	C4
2	6	A6	B6	C6

30. Write a Pandas program to join the two dataframes using the common column of both dataframes

In [33]:
```
import pandas as pd

# Create the first DataFrame
df1 = pd.DataFrame({'id': [1, 2, 3],
                    'A': ['A1', 'A2', 'A3'],
                    'B': ['B1', 'B2', 'B3']})

# Create the second DataFrame
df2 = pd.DataFrame({'id': [2, 3, 4],
                    'C': ['C2', 'C3', 'C4'],
                    'D': ['D2', 'D3', 'D4']})

# Join the two DataFrames using the common column 'id'
result_df = pd.merge(df1, df2, on='id')

print("Resulting DataFrame after joining on 'id':")
result_df
```

```
Resulting DataFrame after joining on 'id':
```

Out[33]:

	id	A	B	C	D
0	2	A2	B2	C2	D2
1	3	A3	B3	C3	D3

31. Write a Pandas program to join (left join) the two dataframes using keys from the left dataframe only

In [30]:
```
import pandas as pd

# Create the left DataFrame
df1 = pd.DataFrame({'key': ['A', 'B', 'C'],
                    'value_left': [1, 2, 3]})

# Create the right DataFrame
df2 = pd.DataFrame({'key': ['A', 'C', 'D'],
                    'value_right': ['X', 'Y', 'Z']})

# Left join the two DataFrames using the 'key' column from the left DataFrame only
result_df = pd.merge(df1, df2, on='key', how='left')

print("Resulting DataFrame after left join:")
result_df
```

```
Resulting DataFrame after left join:
```

Out[30]:

	key	value_left	value_right
0	A	1	X
1	B	2	NaN
2	C	3	Y

32. Write a Pandas program to join two dataframes using keys from the right dataframe only.

In [32]:
```
import pandas as pd

# Create the left DataFrame
df1 = pd.DataFrame({'key': ['A', 'B', 'C'],
                    'value_left': [1, 2, 3]})

# Create the right DataFrame
df2 = pd.DataFrame({'key': ['A', 'C', 'D'],
                    'value_right': ['X', 'Y', 'Z']})

# Right join the two DataFrames using the 'key' column from the right DataFrame only
result_df = pd.merge(df1, df2, on='key', how='right')

print("Resulting DataFrame after right join:")
result_df
```

```
Resulting DataFrame after right join:
```

Out[32]:

	key	value_left	value_right
0	A	1.0	X
1	C	3.0	Y
2	D	NaN	Z

33. Write a Pandas program to merge two given datasets using multiple join keys

In [40]:
```
import pandas as pd

# Create the first DataFrame
df1 = pd.DataFrame({'key1': ['A', 'B', 'C'],
                    'key2': ['X', 'Y', 'Z'],
                    'value1': [1, 2, 3]})
print("df1-----")
print(df1)
# Create the second DataFrame
df2 = pd.DataFrame({'key1': ['A', 'C', 'D'],
                    'key2': ['X', 'Z', 'W'],
                    'value2': ['P', 'Q', 'R']})
print("df2-----")
print(df2)
# Merge the two DataFrames using multiple join keys (key1 and key2)
result_df = pd.merge(df1, df2, on=['key1', 'key2'])
print("-----------------------------------------")
print("Resulting DataFrame after merge with multiple join keys:")
result_df
```

```
df1-----
  key1 key2  value1
0    A    X       1
1    B    Y       2
2    C    Z       3
df2-----
  key1 key2 value2
0    A    X      P
1    C    Z      Q
2    D    W      R
-----------------------------------------
Resulting DataFrame after merge with multiple join keys:
```

Out[40]:

	key1	key2	value1	value2
0	A	X	1	P
1	C	Z	3	Q

34. Write a Pandas program to merge two given dataframes with different columns

In [46]:
```
import pandas as pd
data1 = pd.DataFrame({'key1': ['K0', 'K0', 'K1', 'K2'],
                      'key2': ['K0', 'K1', 'K0', 'K1'],
                      'P': ['P0', 'P1', 'P2', 'P3'],
                      'Q': ['Q0', 'Q1', 'Q2', 'Q3']})
data2 = pd.DataFrame({'key1': ['K0', 'K1', 'K1', 'K2'],
                      'key2': ['K0', 'K0', 'K0', 'K0'],
                      'R': ['R0', 'R1', 'R2', 'R3'],
                      'S': ['S0', 'S1', 'S2', 'S3']})
print("Original DataFrames:")
print(data1)
print("--------------------")
print(data2)
print("\nMerge two dataframes with different columns:")
result = pd.concat([data1,data2], axis=0, ignore_index=True)
result
```

```
Original DataFrames:
  key1 key2   P   Q
0   K0   K0  P0  Q0
1   K0   K1  P1  Q1
2   K1   K0  P2  Q2
3   K2   K1  P3  Q3
--------------------
  key1 key2   R   S
0   K0   K0  R0  S0
1   K1   K0  R1  S1
2   K1   K0  R2  S2
3   K2   K0  R3  S3

Merge two dataframes with different columns:
```

Out[46]:

	key1	key2	P	Q	R	S
0	K0	K0	P0	Q0	NaN	NaN
1	K0	K1	P1	Q1	NaN	NaN
2	K1	K0	P2	Q2	NaN	NaN
3	K2	K1	P3	Q3	NaN	NaN
4	K0	K0	NaN	NaN	R0	S0
5	K1	K0	NaN	NaN	R1	S1
6	K1	K0	NaN	NaN	R2	S2
7	K2	K0	NaN	NaN	R3	S3

35. Write a Pandas program to sort movies on runtime in descending order(Use movies dataset)

In [3]:
```
import pandas as pd
df = pd.read_csv(r"C:\Users\prana\DS - Python\datasets\movies_data.csv")
df1 = df.sort_values(by="duration", ascending=False)
df1
```

Out[3]:

	star_rating	title	content_rating	genre	duration	actors_list
476	7.8	Hamlet	PG-13	Drama	242	[u'Kenneth Branagh', u'Julie Christie', u'Dere...
157	8.2	Gone with the Wind	G	Drama	238	[u'Clark Gable', u'Vivien Leigh', u'Thomas Mit...
78	8.4	Once Upon a Time in America	R	Crime	229	[u'Robert De Niro', u'James Woods', u'Elizabet...
142	8.3	Lagaan: Once Upon a Time in India	PG	Adventure	224	[u'Aamir Khan', u'Gracy Singh', u'Rachel Shell...
445	7.9	The Ten Commandments	APPROVED	Adventure	220	[u'Charlton Heston', u'Yul Brynner', u'Anne Ba...
...	...	...	...	...	...	...
293	8.1	Duck Soup	PASSED	Comedy	68	[u'Groucho Marx', u'Harpo Marx', u'Chico Marx']
88	8.4	The Kid	NOT RATED	Comedy	68	[u'Charles Chaplin', u'Edna Purviance', u'Jack...
258	8.1	The Cabinet of Dr. Caligari	UNRATED	Crime	67	[u'Werner Krauss', u'Conrad Veidt', u'Friedric...
338	8.0	Battleship Potemkin	UNRATED	History	66	[u'Aleksandr Antonov', u'Vladimir Barsky', u'G...
389	8.0	Freaks	UNRATED	Drama	64	[u'Wallace Ford', u'Leila Hyams', u'Olga Bacla...

979 rows × 6 columns

36. Write a Pandas program to get the longest runtime and shortest runtime

In [30]:
```
import pandas as pd
df = pd.read_csv(r"C:\Users\prana\DS - Python\datasets\movies_data.csv")

print("Longest Runtime:",df["duration"].max())
print("Shortest Runtime:",df["duration"].min())
```

```
Longest Runtime: 242
Shortest Runtime: 64
```

In [31]:
```
print("Longest Runtime Whole Row")
pd.DataFrame( df.loc[df['duration'].idxmax()]).T    #df[df['duration'] == df["duration"].max()]   -->  Other Way
```

```
Longest Runtime Whole Row
```

Out[31]:

	star_rating	title	content_rating	genre	duration	actors_list
476	7.8	Hamlet	PG-13	Drama	242	[u'Kenneth Branagh', u'Julie Christie', u'Dere...

In [32]:
```
print("Shortest Runtime Whole Row")
pd.DataFrame( df.loc[df['duration'].idxmin()]).T    #df[df['duration'] == df["duration"].min()]   -->  Other Way
```

```
Shortest Runtime Whole Row
```

Out[32]:

	star_rating	title	content_rating	genre	duration	actors_list
389	8.0	Freaks	UNRATED	Drama	64	[u'Wallace Ford', u'Leila Hyams', u'Olga Bacla...

37. Write a Pandas program to get Action and crime movies

In [36]:
```
import pandas as pd
df = pd.read_csv(r"C:\Users\prana\DS - Python\datasets\movies_data.csv")
print("Dataframe with only Action and Crime movies")
df[ (df['genre'] == 'Action') | (df['genre'] == 'Crime') ]
```

```
Dataframe with only Action and Crime movies
```

Out[36]:

	star_rating	title	content_rating	genre	duration	actors_list
0	9.3	The Shawshank Redemption	R	Crime	142	[u'Tim Robbins', u'Morgan Freeman', u'Bob Gunt...
1	9.2	The Godfather	R	Crime	175	[u'Marlon Brando', u'Al Pacino', u'James Caan']
2	9.1	The Godfather: Part II	R	Crime	200	[u'Al Pacino', u'Robert De Niro', u'Robert Duv...
3	9.0	The Dark Knight	PG-13	Action	152	[u'Christian Bale', u'Heath Ledger', u'Aaron E...
4	8.9	Pulp Fiction	R	Crime	154	[u'John Travolta', u'Uma Thurman', u'Samuel L....
...	...	...	...	...	...	...
963	7.4	La Femme Nikita	R	Action	118	[u'Anne Parillaud', u'Marc Duret', u'Patrick F...
967	7.4	The Rock	R	Action	136	[u'Sean Connery', u'Nicolas Cage', u'Ed Harris']
969	7.4	Law Abiding Citizen	R	Crime	109	[u'Gerard Butler', u'Jamie Foxx', u'Leslie Bibb']
976	7.4	Master and Commander: The Far Side of the World	PG-13	Action	138	[u'Russell Crowe', u'Paul Bettany', u'Billy Bo...
978	7.4	Wall Street	R	Crime	126	[u'Charlie Sheen', u'Michael Douglas', u'Tamar...

260 rows × 6 columns

38. Write a Pandas program to count the city-wise number of people from a given dataset (city, name of the person)

Sample data:

City Number of people

0 California 4

1 Georgia 2

2 Los Angeles 4

In [45]:
```
# Creating Data frame of 30 rows with cities and some random names

import pandas as pd
import random

# Sample city and name data
cities = ['New York', 'Los Angeles', 'Chicago', 'Houston', 'Miami']
names = ['Alice', 'Bob', 'Charlie', 'David', 'Eva', 'Frank', 'Grace', 'Helen', 'Ivy', 'Jack']

# Create a list to store random data
data = []

# Generate 30 random rows of data
for _ in range(30):
    city = random.choice(cities)
    name = random.choice(names)
    data.append({'City': city, 'Name': name})

# Create a DataFrame from the random data
df = pd.DataFrame(data)
```

```
# Display the first few rows of the DataFrame
print("Randomly Generated DataFrame:")
df.head()
```

```
Randomly Generated DataFrame:
```

Out[45]:

	City	Name
0	Los Angeles	Bob
1	Chicago	David
2	New York	David
3	Miami	Bob
4	New York	Charlie

In [47]:
```
# Group by 'City' and count the number of people in each city
city_counts = df.groupby('City')['Name'].count().reset_index()

# Rename the columns for better readability
city_counts.columns = ['City', 'Number of people']

print("City-wise Number of People:")
city_counts
```

```
City-wise Number of People:
```

Out[47]:

	City	Number of people
0	Chicago	6
1	Houston	3
2	Los Angeles	6
3	Miami	6
4	New York	9

39. Write a Pandas program to replace all the NaN values with Zero's in a column of a DataFrame

In [50]:
```
import pandas as pd
import numpy as np

# Creating a DataFrame with NaN values
df = pd.DataFrame({ 'A': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
                    'B': [11, 12, 13, 14, 15, 16, 17, 18, 19, 20],
                    'C': [np.nan, 22, 23, np.nan, 25, 26, np.nan, 28, np.nan, 30]
                  })

# Replace NaN values with zeros in column 'C'
df['C'] = df['C'].fillna(0)

print("DataFrame with NaN values replaced by zeros:")
df
```

```
DataFrame with NaN values replaced by zeros:
```

Out[50]:

	A	B	C
0	1	11	0.0
1	2	12	22.0
2	3	13	23.0
3	4	14	0.0
4	5	15	25.0
5	6	16	26.0
6	7	17	0.0
7	8	18	28.0
8	9	19	0.0
9	10	20	30.0

40. Write a Pandas program to drop a list of rows from a specified DataFrame

Sample data:
Original DataFrame
col1 col2 col3
0 1 4 7
1 4 5 8
2 3 6 9
3 4 7 0
4 5 8 1

In [10]:
```python
import pandas as pd

# Create the original DataFrame
data = {'col1': [1, 4, 3, 4, 5],
        'col2': [4, 5, 6, 7, 8],
        'col3': [7, 8, 9, 0, 1]}

df = pd.DataFrame(data)

# Define a list of row indices to drop
rows_to_drop = [1, 3]

# Use the drop() method to drop the specified rows
df_dropped = df.drop(rows_to_drop)

print("Original DataFrame:")
print(df)

print("\nDataFrame after dropping specified rows:")
print(df_dropped)
```

```
Original DataFrame:
   col1  col2  col3
0     1     4     7
1     4     5     8
2     3     6     9
3     4     7     0
4     5     8     1

DataFrame after dropping specified rows:
   col1  col2  col3
0     1     4     7
2     3     6     9
4     5     8     1
```