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

data = {'A': [1, 2, np.nan, 4, 5],
        'B': [10, np.nan, 30, 40, 50]}
df = pd.DataFrame(data)

mean_A = df['A'].mean()
df['A'] = df['A'].fillna(mean_A)

print(df)

```

```
     A     B
0  1.0  10.0
1  2.0   NaN
2  3.0  30.0
3  4.0  40.0
4  5.0  50.0
```

```
C:\Users\kimay\anaconda3\Lib\site-packages\pandas\core\arrays\masked.py:60: UserWarning: Pandas requires version '1.3.6' or newer of 'bottleneck' (version '1.3.5' currently installed).
  from pandas.core import (
```

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

In [2]:
```
df = {'NAME': ['KIMAYA','RAHUL','EKTA','MAYANK'],
      'AGE': [20,21,22,19],
      'CITY': ['MUMBAI','NOIDA','PUNE','DELHI']
}

INDEX_LABELS =['A','B','C','D']
df=pd.DataFrame(df,INDEX_LABELS)
print(df)
```

```
     NAME  AGE    CITY
A  KIMAYA   20  MUMBAI
B   RAHUL   21   NOIDA
C    EKTA   22    PUNE
D  MAYANK   19   DELHI
```

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

In [3]:
```
df = {'NAME': ['KIMAYA','RAHUL','EKTA','MAYANK'],
      'AGE': [20,21,22,19],
      'CITY': ['MUMBAI','NOIDA','PUNE','DELHI']
}
df=pd.DataFrame(df)
rows = df.head(3)
print(rows)

```

```
    NAME  AGE    CITY
0  KIMAYA   20  MUMBAI
1   RAHUL   21   NOIDA
2    EKTA   22    PUNE
```

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

In [4]:
```
df = {'NAME': ['KIMAYA','RAHUL','EKTA','MAYANK','YASH'],
      'AGE': [20,21,22,19,21],
      'CITY': ['MUMBAI','NOIDA','PUNE','DELHI','BANGALORE ']
}
df=pd.DataFrame(df)

rows = df.iloc[:2, :2]
print("First 2 Rows and 2 Columns:")
print(rows)

specific_columns = df[['NAME','CITY']]
print("\nSpecific 2 Clumns:")
print(specific_columns)
```

```
First 2 Rows and 2 Columns:
    NAME  AGE
0  KIMAYA   20
1   RAHUL   21

Specific 2 Clumns:
    NAME        CITY
0  KIMAYA      MUMBAI
1   RAHUL       NOIDA
2    EKTA        PUNE
3  MAYANK       DELHI
4    YASH  BANGALORE
```

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

In [5]:
```
df = {'NAME': ['KIMAYA','RAHUL','EKTA','MAYANK','YASH'],
      'AGE': [20,21,22,19,21],
      'CITY': ['MUMBAI','NOIDA','PUNE','DELHI','BANGALORE ']
}
df=pd.DataFrame(df)

A = df.loc[[1,4],['NAME','AGE']]
print("specified columns and rows:")
print(A)

```

```
specified columns and rows:
    NAME  AGE
1  RAHUL   21
4   YASH   21
```

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

In [6]:
```
df = {'NAME': ['KIMAYA','RAHUL','EKTA','MAYANK','YASH'],
      'AGE': [20,21,np.nan,19,21],
      'CITY': ['MUMBAI','NOIDA','PUNE','DELHI',np.nan]
}
df=pd.DataFrame(df)

Missing_value = df.isnull()
print(Missing_value)
```

```
    NAME    AGE   CITY
0  False  False  False
1  False  False  False
2  False   True  False
3  False  False  False
4  False  False   True
```

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

In [7]:
```
df = {'NAME': ['KIMAYA','RAHUL','EKTA','MAYANK','YASH'],
      'AGE': [20,21,np.nan,19,21],
      'CITY': ['MUMBAI','NOIDA','PUNE','DELHI',np.nan]
}
df=pd.DataFrame(df)

Missing_value = df.isnull().sum()
print(Missing_value)
```

```
NAME    0
AGE     1
CITY    1
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 [8]:
```
df = {'NAME': ['KIMAYA','RAHUL','EKTA','MAYANK','--'],
      'AGE': [20,21,'?',19,21],
      'CITY': ['?','NOIDA','PUNE','DELHI','--']
}
df=pd.DataFrame(df)

Missing_value = ['?','--']
df.replace(Missing_value, np.NaN, inplace=True)
print(df)
```

```
     NAME   AGE   CITY
0  KIMAYA  20.0    NaN
1   RAHUL  21.0  NOIDA
2    EKTA   NaN   PUNE
3  MAYANK  19.0  DELHI
4     NaN  21.0    NaN

C:\Users\kimay\AppData\Local\Temp\ipykernel_3736\3855235949.py:8: FutureWarning: Downcasting behavior in `replace` is deprecated and will be removed in a future version. To retain the old behavior, explicitly call `result.infer_objects(copy=False)`. To opt-in to the future behavior, set `pd.set_option('future.no_silent_downcasting', True)`
  df.replace(Missing_value, np.NaN, inplace=True)
```

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

In [9]:
```
data = {
    'NAME': ['KIMAYA', 'RAHUL', 'EKTA', 'MAYANK', 'YASH'],
    'AGE': [20, 21, np.nan, 19, 21],
    'CITY': ['MUMBAI', 'NOIDA', 'PUNE', 'DELHI', np.nan]
}
df = pd.DataFrame(data)

df_cleaned = df.dropna()
print(df_cleaned )
```

```
     NAME   AGE    CITY
0  KIMAYA  20.0  MUMBAI
1   RAHUL  21.0   NOIDA
3  MAYANK  19.0   DELHI
```

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

In [10]:
```
df = {
    'NAME': ['KIMAYA', 'RAHUL', 'EKTA', 'MAYANK', '--'],
    'AGE': [20, 21, np.nan, 19, 21],
    'CITY': ['MUMBAI', 'NOIDA', 'PUNE', 'DELHI', np.nan]
}
df = pd.DataFrame(df)

df.replace('--', np.nan, inplace=True)
valid_entries = df.dropna(how='any')
print(valid_entries)
```

```
     NAME   AGE    CITY
0  KIMAYA  20.0  MUMBAI
1   RAHUL  21.0   NOIDA
3  MAYANK  19.0   DELHI
```

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

In [11]:
```
df = {
    'NAME': ['KIMAYA', 'RAHUL', np.nan, 'MAYANK', 'YASH'],
    'AGE': [20, 21, np.nan, 19, 21],
    'CITY': ['MUMBAI', 'NOIDA', 'PUNE', np.nan, np.nan]
}
df = pd.DataFrame(df)

total_missing = df.isnull().sum().sum()
print("Total number of missing values in a DataFrame:")
print(total_missing)
```

```
Total number of missing values in a DataFrame:
4
```

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

In [12]:
```
data = {
    'NAME': ['KIMAYA', 'RAHUL', np.nan, 'MAYANK', 'YASH'],
    'AGE': [20, 21, np.nan, 19, 21],
    'CITY': ['MUMBAI', 'NOIDA', 'PUNE', np.nan, 'CHENNAI']
}
df = pd.DataFrame(data)

column = ['NAME', 'AGE', 'CITY']
constant_value = 'XXXXX'
df[column] = df[column].fillna(constant_value)
print(df)
```

```
     NAME    AGE     CITY
0  KIMAYA   20.0   MUMBAI
1   RAHUL   21.0    NOIDA
2   XXXXX  XXXXX     PUNE
3  MAYANK   19.0    XXXXX
4    YASH   21.0  CHENNAI
```

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

In [13]:
```
data = {
    'NAME': ['KIMAYA', 'RAHUL', np.nan, 'MAYANK', 'YASH'],
    'AGE': [20, 21, np.nan, 19, 21],
    'CITY': ['MUMBAI', 'NOIDA', 'PUNE', np.nan, 'CHENNAI']
}
df = pd.DataFrame(data)

column = ['AGE']
for col in column:
    median_value = df[col].median()
    mean_value = df[col].mean()
    df[col].fillna(median_value, inplace=True)
    df[col].fillna(mean_value, inplace=True)
print("Replace NaNs with the median or mean value:")
print(df)
```

```
Replace NaNs with the median or mean value:
     NAME   AGE     CITY
0  KIMAYA  20.0   MUMBAI
1   RAHUL  21.0    NOIDA
2     NaN  20.5     PUNE
3  MAYANK  19.0      NaN
4    YASH  21.0  CHENNAI

C:\Users\kimay\AppData\Local\Temp\ipykernel_3736\854433415.py:12: FutureWa
rning: A value is trying to be set on a copy of a DataFrame or Series thro
ugh chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never wor
k because the intermediate object on which we are setting values always be
haves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using
'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value)
instead, to perform the operation inplace on the original object.


  df[col].fillna(median_value, inplace=True)
C:\Users\kimay\AppData\Local\Temp\ipykernel_3736\854433415.py:13: FutureWa
rning: A value is trying to be set on a copy of a DataFrame or Series thro
ugh chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never wor
k because the intermediate object on which we are setting values always be
haves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using
'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value)
instead, to perform the operation inplace on the original object.


  df[col].fillna(mean_value, inplace=True)
```

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

In [14]:
```
data = {
    'NAME': ['KIMAYA', 'RAHUL', np.nan, 'MAYANK', 'YASH'],
    'AGE': [20, 21, np.nan, 19, 21],
    'CITY': ['MUMBAI', 'NOIDA', 'PUNE', np.nan, 'CHENNAI']
}
df = pd.DataFrame(data)

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

```

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

15. Write a Pandas program to select rows from a given DataFrame ba sed on values in some columns

In [3]:
```
import pandas as pd
data = {
    'NAME': ['KIMAYA', 'RAHUL', 'EKTA', 'MAYANK', 'YASH'],
    'AGE': [20, 21, 25, 19, 21],
    'CITY': ['MUMBAI', 'NOIDA', 'PUNE', 'DELHI', 'CHENNAI']
}
df = pd.DataFrame(data)

selected_rows = df[(df['AGE'] >= 20) & (df['CITY'] == 'PUNE')]

print("Selected rows based on conditions:")
print(selected_rows)
```

```
C:\Users\kimay\anaconda3\Lib\site-packages\pandas\core\arrays\masked.py:6
0: UserWarning: Pandas requires version '1.3.6' or newer of 'bottleneck'
(version '1.3.5' currently installed).
  from pandas.core import (

Selected rows based on conditions:
   NAME  AGE  CITY
2  EKTA   25  PUNE
```

16. Write a Pandas program to change the order of a DataFrame colum ns

In [16]:

```
data = {
    'NAME': ['KIMAYA', 'RAHUL', 'EKTA', 'MAYANK', 'YASH'],
    'AGE': [20, 21, 25, 19, 21],
    'CITY': ['MUMBAI', 'NOIDA', 'PUNE', 'DELHI', 'CHENNAI']
}
df = pd.DataFrame(data)

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

df = df[['AGE', 'NAME', 'CITY']]
print("\nDataFrame with changed column order:")
print(df)

```

```
Original DataFrame:
     NAME  AGE     CITY
0  KIMAYA   20   MUMBAI
1   RAHUL   21    NOIDA
2    EKTA   25     PUNE
3  MAYANK   19    DELHI
4    YASH   21  CHENNAI

DataFrame with changed column order:
   AGE    NAME     CITY
0   20  KIMAYA   MUMBAI
1   21   RAHUL    NOIDA
2   25    EKTA     PUNE
3   19  MAYANK    DELHI
4   21    YASH  CHENNAI
```

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

In [17]:

```
data = {
    'NAME': ['KIMAYA', 'RAHUL', 'EKTA', 'MAYANK', 'YASH'],
    'AGE': [20, 21, 25, 19, 21],
    'CITY': ['MUMBAI', 'NOIDA', 'PUNE', 'DELHI', 'CHENNAI']
}
df = pd.DataFrame(data)


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


new_data = {
    'NAME': 'ANISHA',
    'AGE': 22,
    'CITY': 'BENGALURU'
}

df = df.append(new_data, ignore_index=True)
print("\nDataFrame with added row:")
print(df)

```

```
Original DataFrame:
     NAME  AGE     CITY
0  KIMAYA   20   MUMBAI
1   RAHUL   21    NOIDA
2    EKTA   25     PUNE
3  MAYANK   19    DELHI
4    YASH   21  CHENNAI

---------------------------------------------------------------------------
-
AttributeError                            Traceback (most recent call las
t)
~\AppData\Local\Temp\ipykernel_3736\412281069.py in ?()
     15     'AGE': 22,
     16     'CITY': 'BENGALURU'
     17 }
     18
---> 19 df = df.append(new_data, ignore_index=True)
     20 print("\nDataFrame with added row:")
     21 print(df)

~\anaconda3\Lib\site-packages\pandas\core\generic.py in ?(self, name)
   6292             and name not in self._accessors
   6293             and self._info_axis._can_hold_identifiers_and_holds_na
me(name)
   6294         ):
   6295             return self[name]
-> 6296         return object.__getattribute__(self, name)

AttributeError: 'DataFrame' object has no attribute 'append'
```

18. Write a Pandas program to delete DataFrame row(s) based on a gi ven column value

In [18]:
```
data = {
    'NAME': ['KIMAYA', 'RAHUL', 'EKTA', 'MAYANK', 'YASH'],
    'AGE': [20, 21, 25, 19, 21],
    'CITY': ['MUMBAI', 'NOIDA', 'PUNE', 'DELHI', 'CHENNAI']
}
df = pd.DataFrame(data)

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

df = df[df['CITY'] != 'DELHI']
print("\nDataFrame after deleting rows where CITY is 'DELHI':")
print(df)

```

```
Original DataFrame:
     NAME  AGE     CITY
0  KIMAYA   20   MUMBAI
1   RAHUL   21    NOIDA
2    EKTA   25     PUNE
3  MAYANK   19    DELHI
4    YASH   21  CHENNAI

DataFrame after deleting rows where CITY is 'DELHI':
     NAME  AGE     CITY
0  KIMAYA   20   MUMBAI
1   RAHUL   21    NOIDA
2    EKTA   25     PUNE
4    YASH   21  CHENNAI
```

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

In [19]:
```
data = {
    'NAME': ['KIMAYA', 'RAHUL', 'EKTA', 'MAYANK', 'YASH'],
    'AGE': [20, 21, 25, 19, 21],
    'CITY': ['MUMBAI', 'NOIDA', 'PUNE', 'DELHI', 'CHENNAI']
}
df = pd.DataFrame(data)
print("Original DataFrame:")
print(df)


selected_row = df.iloc[2]
print("\nSelected row at index 2:")
print(selected_row)

```

```
Original DataFrame:
     NAME  AGE     CITY
0  KIMAYA   20   MUMBAI
1   RAHUL   21    NOIDA
2    EKTA   25     PUNE
3  MAYANK   19    DELHI
4    YASH   21  CHENNAI

Selected row at index 2:
NAME    EKTA
AGE       25
CITY    PUNE
Name: 2, dtype: object
```

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

In [20]:
```
data = {
    'NAME': ['KIMAYA', 'RAHUL', 'EKTA', 'MAYANK', 'YASH'],
    'AGE': [20, 21, 25, 19, 21],
    'CITY': ['MUMBAI', 'NOIDA', 'PUNE', 'DELHI', 'CHENNAI']
}
df = pd.DataFrame(data)

df['NAME_LENGTH'] = df['NAME'].str.len()

print(df)
```

```
     NAME  AGE     CITY  NAME_LENGTH
0  KIMAYA   20   MUMBAI            6
1   RAHUL   21    NOIDA            5
2    EKTA   25     PUNE            4
3  MAYANK   19    DELHI            6
4    YASH   21  CHENNAI            4
```

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

In [22]:
```
data = {
    'NAME': ['KIMAYA', 'RAHUL', 'EKTA', 'MAYANK', 'YASH'],
    'AGE': [20, 21, 25, 19, 21],
    'CITY': ['MUMBAI', 'NOIDA', 'PUNE', 'DELHI', 'CHENNAI']
}
df = pd.DataFrame(data)

column_to_swap = 'NAME'
df[column_to_swap] = df[column_to_swap].str.swapcase()
print("Updated DataFrame:")
print(df)

```

```
Updated DataFrame:
     NAME  AGE     CITY
0  kimaya   20   MUMBAI
1   rahul   21    NOIDA
2    ekta   25     PUNE
3  mayank   19    DELHI
4    yash   21  CHENNAI
```

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

In [23]:
```
data = {
    'NAME': ['KIMAYA', 'RAHUL', 'EKTA', 'MAYANK', 'YASH'],
    'AGE': [20, 21, 25, 19, 21],
    'CITY': ['MUMBAI', 'NOIDA', 'PUNE', 'DELHI', 'CHENNAI']
}
df = pd.DataFrame(data)

column = 'NAME'
df[column] = df[column].str.lower()
print("Lower case")
print(df)
df[column] = df[column].str.upper()
print("\nUpper case:")
print(df)
```

```
Lower case
     NAME  AGE     CITY
0  kimaya   20   MUMBAI
1   rahul   21    NOIDA
2    ekta   25     PUNE
3  mayank   19    DELHI
4    yash   21  CHENNAI

Upper case:
     NAME  AGE     CITY
0  KIMAYA   20   MUMBAI
1   RAHUL   21    NOIDA
2    EKTA   25     PUNE
3  MAYANK   19    DELHI
4    YASH   21  CHENNAI
```

23. Write a Pandas program to remove whitespaces, left-sided whites paces, and right-sided whitespaces of the string values of a gi ven pandas series

In [27]:
```
import pandas as pd
data = { 'NAME': ['  KIMAYA', '  RAHUL  ', 'EKTA   ', '   MAYANK', '  Y
df = pd.DataFrame(data)

column = 'NAME'
All_space = df[column].str.replace(" ","")

Left_side = df[column].str.lstrip()

Right_side = df[column].str.rstrip()

print("origanal dataframe:")
print(column)
print("\nall remove whitespace:")
print(All_space)
print("\nLeft-sided whitespace:")
print(Left_side)
print("\nRight-sided whitespace:")
print(Right_side)
```

```
origanal dataframe:
NAME

all remove whitespace:
0    KIMAYA
1     RAHUL
2      EKTA
3    MAYANK
4      YASH
Name: NAME, dtype: object

Left-sided whitespace:
0     KIMAYA
1    RAHUL
2    EKTA
3     MAYANK
4     YASH
Name: NAME, dtype: object

Right-sided whitespace:
0       KIMAYA
1        RAHUL
2         EKTA
3       MAYANK
4         YASH
Name: NAME, dtype: object
```

24. Write a Pandas program to join the two given dataframes along r ows

In [34]:
```
df1 = pd.DataFrame({
    'A': ['A1', 'A2', 'A3'],
    'B': ['B1', 'B2', 'B3'],
})

df2 = pd.DataFrame({
    'A': ['A4', 'A5', 'A6'],
    'B': ['B4', 'B5', 'B6'],
})

result_df = pd.concat([df1, df2])

print("Concatenated DataFrame:")
print(result_df)

```

```
Concatenated DataFrame:
    A   B
0  A1  B1
1  A2  B2
2  A3  B3
0  A4  B4
1  A5  B5
2  A6  B6
```

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

In [35]:
```
df1 = pd.DataFrame({
    'A': ['A1', 'A2', 'A3'],
    'B': ['B1', 'B2', 'B3'],
})

df2 = pd.DataFrame({
    'C': ['C4', 'C5', 'C6'],
    'D': ['D4', 'D5', 'D6'],
})

result_df = pd.concat([df1, df2], axis=1)

print("Concatenated DataFrame:")
print(result_df)
```

```
Concatenated DataFrame:
    A   B   C   D
0  A1  B1  C4  D4
1  A2  B2  C5  D5
2  A3  B3  C6  D6
```

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

In [36]:
```
df1 = pd.DataFrame({
    'id': [1, 2, 3],
    'A': ['A1', 'A2', 'A3'],
    'B': ['B1', 'B2', 'B3'],
})

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

df3 = pd.DataFrame({
    'id': [1, 2, 3, 4, 5, 6],
    'C': ['C1', 'C2', 'C3', 'C4', 'C5', 'C6'],
    'D': ['D1', 'D2', 'D3', 'D4', 'D5', 'D6'],
})

concatenated_df = pd.concat([df1, df2], ignore_index=True)
merged_df = pd.merge(concatenated_df, df3, on='id')

print("Concatenated and Merged DataFrame:")
print(merged_df)

```

```
Concatenated and Merged DataFrame:
   id   A   B   C   D
0   1  A1  B1  C1  D1
1   2  A2  B2  C2  D2
2   3  A3  B3  C3  D3
3   4  A4  B4  C4  D4
4   5  A5  B5  C5  D5
5   6  A6  B6  C6  D6
```

27. Write a Pandas program to join the two dataframes using the com mon column of both dataframes

In [41]:
```
df1 = pd.DataFrame({
    'id': [1, 2, 3],
    'A': ['A1', 'A2', 'A3'],
    'B': ['B1', 'B2', 'B3'],
})

df2 = pd.DataFrame({
    'id': [1, 2, 3],
    'C': ['A4', 'A5', 'A6'],
    'D': ['B4', 'B5', 'B6'],
})
marged_df = pd.merge(df1, df2, on='id')
print("merege two dataframe")
print(marged_df)
```

```
merege two dataframe
   id   A   B   C   D
0   1  A1  B1  A4  B4
1   2  A2  B2  A5  B5
2   3  A3  B3  A6  B6
```

28. Write a Pandas program to join (left join) the two dataframes u sing keys from the left dataframe only

In [44]:
```
df1 = pd.DataFrame({
    'id': [1, 2, 3],
    'A': ['A1', 'A2', 'A3'],
    'B': ['B1', 'B2', 'B3'],
})

df2 = pd.DataFrame({
    'id': [2, 3, 4],
    'C': ['A4', 'A5', 'A6'],
    'D': ['B4', 'B5', 'B6'],
})
DF = pd.merge(df1, df2, on='id', how='left')
print("Print only left dataframe:")
print(DF)
```

```
Print only left dataframe:
   id   A   B    C    D
0   1  A1  B1  NaN  NaN
1   2  A2  B2   A4   B4
2   3  A3  B3   A5   B5
```

29. Write a Pandas program to join two dataframes using keys from t he right dataframe only.

In [47]:
```
df1 = pd.DataFrame({
    'id': [1, 2, 3],
    'A': ['A1', 'A2', 'A3'],
    'B': ['B1', 'B2', 'B3'],
})

df2 = pd.DataFrame({
    'id': [2, 3, 4],
    'C': ['A4', 'A5', 'A6'],
    'D': ['B4', 'B5', 'B6'],
})
DF = pd.merge(df1, df2, on='id', how='right')
print("Print only Right dataframe:")
print(DF)
```

```
Print only Right dataframe:
   id    A    B   C   D
0   2   A2   B2  A4  B4
1   3   A3   B3  A5  B5
2   4  NaN  NaN  A6  B6
```

30. Write a Pandas program to merge two given datasets using multip le join keys

In [48]:
```
df1 = pd.DataFrame({
    'key1': ['K0', 'K1', 'K2', 'K3'],
    'key2': ['A', 'B', 'C', 'D'],
    'A': ['A1', 'A2', 'A3', 'A4'],
    'B': ['B1', 'B2', 'B3', 'B4'],
})

df2 = pd.DataFrame({
    'key1': ['K0', 'K1', 'K2', 'K3'],
    'key2': ['A', 'B', 'C', 'D'],
    'C': ['C1', 'C2', 'C3', 'C4'],
    'D': ['D1', 'D2', 'D3', 'D4'],
})

merged_df = pd.merge(df1, df2, on=['key1', 'key2'])
print("Merged DataFrame using multiple join keys:")
print(merged_df)

```

```
Merged DataFrame using multiple join keys:
  key1 key2   A   B   C   D
0   K0    A  A1  B1  C1  D1
1   K1    B  A2  B2  C2  D2
2   K2    C  A3  B3  C3  D3
3   K3    D  A4  B4  C4  D4
```

31. Write a Pandas program to merge two given dataframes with diffe rent columns

In [49]:
```
df1 = pd.DataFrame({
    'id': [1, 2, 3, 4],
    'A': ['A1', 'A2', 'A3', 'A4'],
    'B': ['B1', 'B2', 'B3', 'B4'],
})

df2 = pd.DataFrame({
    'id': [3, 4, 5, 6],
    'C': ['C3', 'C4', 'C5', 'C6'],
    'D': ['D3', 'D4', 'D5', 'D6'],
})

merged_df = pd.merge(df1, df2, on='id', how='outer')

print("Merged DataFrame with different columns:")
print(merged_df)

```

```
Merged DataFrame with different columns:
   id    A    B    C    D
0   1   A1   B1  NaN  NaN
1   2   A2   B2  NaN  NaN
2   3   A3   B3   C3   D3
3   4   A4   B4   C4   D4
4   5  NaN  NaN   C5   D5
5   6  NaN  NaN   C6   D6
```

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

In [51]:
```
data = {
    'title': ['Jawan', 'Animal', 'Pathaan', 'Gadar2'],
    'runtime': [120, 150, 90, 110]
}
movies_df = pd.DataFrame(data)

movies = movies_df.sort_values(by='runtime', ascending=False)
print("Movies sorted by runtime in descending order:")
print(movies_df)

```

```
Movies sorted by runtime in descending order:
     title  runtime
0    Jawan      120
1   Animal      150
2  Pathaan       90
3   Gadar2      110
```

33. Write a Pandas program to replace all the NaN values with Zer o's in a column of a DataFrame

In [52]:
```
data = {
    'A': [1, 2, np.nan, 4, 5, np.nan, 7],
    'B': [10, np.nan, 30, np.nan, 50, 60, np.nan]
}
df = pd.DataFrame(data)

df['A'] = df['A'].fillna(0)
df['B'] = df['B'].fillna(0)
print(df)

```

```
     A     B
0  1.0  10.0
1  2.0   0.0
2  0.0  30.0
3  4.0   0.0
4  5.0  50.0
5  0.0  60.0
6  7.0   0.0
```

34. 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 [54]:

```
data = {
    'col1': [1, 4, 3, 4, 5],
    'col2': [4, 5, 6, 7, 8],
    'col3': [7, 8, 9, 0, 1]
}
df = pd.DataFrame(data)

rows = [1, 3]

df.drop(rows, inplace=True)
print(df)

```

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