pandas-p1

February 13, 2025

[1]:
```
# Program 1: Create a DataFrame from a Dictionary
import pandas as pd

data = {'Name': ['John', 'Anna', 'Peter'], 'Age': [28, 24, 35]}
df = pd.DataFrame(data)
print(df)
```

```
    Name  Age
0   John   28
1   Anna   24
2  Peter   35
```

[2]:
```
# Program 2: Display the Shape of a DataFrame
import pandas as pd

df = pd.DataFrame({'Name': ['John', 'Anna', 'Peter'], 'Age': [28, 24, 35]})
print(df.shape)  # Returns the number of rows and columns
```

```
(3, 2)
```

[3]:
```
# Program 3: Display Column Names
import pandas as pd

df = pd.DataFrame({'Name': ['John', 'Anna', 'Peter'], 'Age': [28, 24, 35]})
print(df.columns)
```

```
Index(['Name', 'Age'], dtype='object')
```

[4]:
```
# Program 4: Select Specific Columns
import pandas as pd

df = pd.DataFrame({'Name': ['John', 'Anna', 'Peter'], 'Age': [28, 24, 35],␣
  ↪'City': ['NY', 'LA', 'SF']})
print(df[['Name', 'Age']])
```

```
    Name  Age
0   John   28
```

```
1   Anna   24
2  Peter   35
```

[5]:
```python
# Program 5: Filter Rows Based on Condition
import pandas as pd

df = pd.DataFrame({'Name': ['John', 'Anna', 'Peter'], 'Age': [28, 24, 35]})
print(df[df['Age'] > 30])
```

```
    Name  Age
2  Peter   35
```

[6]:
```python
# Program 6: Sort DataFrame by a Column
import pandas as pd

df = pd.DataFrame({'Name': ['John', 'Anna', 'Peter'], 'Age': [28, 24, 35]})
print(df.sort_values(by='Age'))
```

```
    Name  Age
1   Anna   24
0   John   28
2  Peter   35
```

[7]:
```python
# Program 7: Add a New Column
import pandas as pd

df = pd.DataFrame({'Name': ['John', 'Anna', 'Peter'], 'Age': [28, 24, 35]})
df['City'] = ['NY', 'LA', 'SF']
print(df)
```

```
    Name  Age City
0   John   28   NY
1   Anna   24   LA
2  Peter   35   SF
```

[8]:
```python
# Program 8: Drop a Column
import pandas as pd

df = pd.DataFrame({'Name': ['John', 'Anna', 'Peter'], 'Age': [28, 24, 35],␣
 ↪'City': ['NY', 'LA', 'SF']})
df = df.drop('City', axis=1)
print(df)
```

```
    Name  Age
0   John   28
1   Anna   24
2  Peter   35
```

[9]:
```
# Program 9: Rename a Column
import pandas as pd

df = pd.DataFrame({'Name': ['John', 'Anna', 'Peter'], 'Age': [28, 24, 35]})
df.rename(columns={'Age': 'Years'}, inplace=True)
print(df)
```

```
    Name  Years
0   John     28
1   Anna     24
2  Peter     35
```

[10]:
```
# Program 10: Get the First Few Rows
import pandas as pd

df = pd.DataFrame({'Name': ['John', 'Anna', 'Peter'], 'Age': [28, 24, 35]})
print(df.head(2))  # Show the first 2 rows
```

```
   Name  Age
0  John   28
1  Anna   24
```

[11]:
```
# Program 11: Get the Last Few Rows
import pandas as pd

df = pd.DataFrame({'Name': ['John', 'Anna', 'Peter'], 'Age': [28, 24, 35]})
print(df.tail(2))  # Show the last 2 rows
```

```
    Name  Age
1   Anna   24
2  Peter   35
```

[12]:
```
# Program 12: Check for Missing Values
import pandas as pd

df = pd.DataFrame({'Name': ['John', 'Anna', 'Peter'], 'Age': [28, None, 35]})
print(df.isnull())
```

```
    Name    Age
0  False  False
1  False   True
2  False  False
```

[13]:
```
# Program 13: Fill Missing Values with a Default Value
import pandas as pd

df = pd.DataFrame({'Name': ['John', 'Anna', 'Peter'], 'Age': [28, None, 35]})
df['Age'].fillna(30, inplace=True)
```

```python
print(df)
```

```
    Name   Age
0   John  28.0
1   Anna  30.0
2  Peter  35.0
```

```
C:\Users\Ravi Kumar Verma\AppData\Local\Temp\ipykernel_17980\964532615.py:5:
FutureWarning: A value is trying to be set on a copy of a DataFrame or Series
through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work
because the intermediate object on which we are setting values always behaves 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['Age'].fillna(30, inplace=True)
```

[14]:
```python
# Program 14: Count Unique Values in a Column
import pandas as pd

df = pd.DataFrame({'Name': ['John', 'Anna', 'Peter', 'John'], 'Age': [28, 24,␣
 ↪35, 28]})
print(df['Name'].nunique())
```

```
3
```

[15]:
```python
# Program 15: Get the Value Counts of a Column
import pandas as pd

df = pd.DataFrame({'Name': ['John', 'Anna', 'Peter', 'John'], 'Age': [28, 24,␣
 ↪35, 28]})
print(df['Name'].value_counts())
```

```
Name
John     2
Anna     1
Peter    1
Name: count, dtype: int64
```

[16]:
```python
# Program 16: Check if a Column Exists
import pandas as pd

df = pd.DataFrame({'Name': ['John', 'Anna', 'Peter'], 'Age': [28, 24, 35]})
print('Age' in df.columns)  # Check if 'Age' column exists
```

```
True
```

[17]:
```python
# Program 17: Find the Maximum Value in a Column
import pandas as pd

df = pd.DataFrame({'Name': ['John', 'Anna', 'Peter'], 'Age': [28, 24, 35]})
print(df['Age'].max())  # Get the maximum value in 'Age' column
```

```
35
```

[18]:
```python
# Program 18: Find the Minimum Value in a Column
import pandas as pd

df = pd.DataFrame({'Name': ['John', 'Anna', 'Peter'], 'Age': [28, 24, 35]})
print(df['Age'].min())  # Get the minimum value in 'Age' column
```

```
24
```

[19]:
```python
# Program 19: Filter Rows Based on Multiple Conditions
import pandas as pd

df = pd.DataFrame({'Name': ['John', 'Anna', 'Peter'], 'Age': [28, 24, 35],␣
 ↪'City': ['NY', 'LA', 'SF']})
print(df[(df['Age'] > 25) & (df['City'] == 'NY')])
```

```
   Name  Age City
0  John   28   NY
```

[20]:
```python
# Program 20: Change Data Type of a Column
import pandas as pd

df = pd.DataFrame({'Name': ['John', 'Anna', 'Peter'], 'Age': [28, 24, 35]})
df['Age'] = df['Age'].astype('float64')
print(df.dtypes)
```

```
Name     object
Age     float64
dtype: object
```

[21]:
```python
# Program 1: Merge Two DataFrames on a Common Column
import pandas as pd

# Create two DataFrames
df1 = pd.DataFrame({'ID': [1, 2, 3, 4], 'Name': ['John', 'Anna', 'Peter',␣
 ↪'Linda'], 'Age': [28, 24, 35, 33]})
df2 = pd.DataFrame({'ID': [2, 3, 4, 5], 'City': ['LA', 'SF', 'NY', 'LA']})

# Merge the DataFrames on the 'ID' column
```

```
merged_df = pd.merge(df1, df2, on='ID', how='inner')

# Display the result
print(merged_df)
```

```
   ID   Name  Age City
0   2   Anna   24   LA
1   3  Peter   35   SF
2   4  Linda   33   NY
```

[22]:
```
# Program 2: Group Data by a Column and Calculate Aggregations
import pandas as pd

# Create a DataFrame
df = pd.DataFrame({'Name': ['John', 'Anna', 'Peter', 'John', 'Anna', 'Peter'],
                   'Age': [28, 24, 35, 28, 24, 35],
                   'City': ['NY', 'LA', 'SF', 'NY', 'LA', 'SF'],
                   'Salary': [50000, 60000, 55000, 48000, 62000, 54000]})

# Group by 'Name' and calculate the average age and total salary
grouped_df = df.groupby('Name').agg({'Age': 'mean', 'Salary': 'sum'})

# Display the result
print(grouped_df)
```

```
        Age  Salary
Name
Anna   24.0  122000
John   28.0   98000
Peter  35.0  109000
```

[23]:
```
# Program 3: Pivot Table to Summarize Data
import pandas as pd

# Create a DataFrame
df = pd.DataFrame({'Date': ['2024-01-01', '2024-01-02', '2024-01-03',␣
 ↪'2024-01-04'],
                   'City': ['NY', 'LA', 'SF', 'NY'],
                   'Sales': [200, 220, 150, 180]})

# Create a pivot table with Date as the index, City as the columns, and Sales␣
 ↪as the values
pivot_table = df.pivot_table(values='Sales', index='Date', columns='City',␣
 ↪aggfunc='sum')

# Display the result
print(pivot_table)
```

```
City            LA     NY     SF
Date
2024-01-01     NaN  200.0    NaN
2024-01-02   220.0    NaN    NaN
2024-01-03     NaN    NaN  150.0
2024-01-04     NaN  180.0    NaN
```

[24]:
```
# Program 4: Apply a Function to Each Row Using `apply`
import pandas as pd

# Create a DataFrame
df = pd.DataFrame({'Name': ['John', 'Anna', 'Peter', 'Linda'],
                   'Age': [28, 24, 35, 33],
                   'City': ['NY', 'LA', 'SF', 'NY']})

# Define a custom function to classify people by age group
def classify_age_group(row):
    if row['Age'] < 30:
        return 'Young'
    elif row['Age'] < 40:
        return 'Adult'
    else:
        return 'Senior'

# Apply the function to each row and create a new column 'Age Group'
df['Age Group'] = df.apply(classify_age_group, axis=1)

# Display the result
print(df)
```

```
    Name  Age City Age Group
0   John   28   NY     Young
1   Anna   24   LA     Young
2  Peter   35   SF     Adult
3  Linda   33   NY     Adult
```

[25]:
```
# Program 5: Read, Modify, and Save a DataFrame to CSV
import pandas as pd

# Create a DataFrame
df = pd.DataFrame({'Name': ['John', 'Anna', 'Peter', 'Linda'],
                   'Age': [28, 24, 35, 33],
                   'City': ['NY', 'LA', 'SF', 'NY']})

# Save the DataFrame to a CSV file
df.to_csv('people.csv', index=False)
```

```python
# Read the DataFrame from the CSV file
df_read = pd.read_csv('people.csv')

# Modify the DataFrame
df_read['Age'] = df_read['Age'] + 1  # Increment the age by 1 year

# Save the modified DataFrame back to a new CSV file
df_read.to_csv('people_modified.csv', index=False)

# Display the modified DataFrame
print(df_read)
```

```
    Name  Age City
0   John   29   NY
1   Anna   25   LA
2  Peter   36   SF
3  Linda   34   NY
```

```
[ ]:
```