Experiment12

April 8, 2025

Name - Shravani Sandeep Raut

SE - 48

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

[13]:
```
# Step 1: Creating a DataFrame using a dictionary
data_dict = {
    'Name': ['Alice', 'Bob', 'Charlie', 'David', 'Eva'],
    'Age': [23, 30, 35, 40, 28],
    'City': ['New York', 'Los Angeles', 'Chicago', 'Houston', 'Phoenix'],
    'Salary': [70000, 80000, 120000, 100000, 95000]
}

df_from_dict = pd.DataFrame(data_dict)
```

[14]:
```
# Step 2: Display the DataFrame
print("DataFrame from Dictionary:")
print(df_from_dict)
```

```
DataFrame from Dictionary:
      Name  Age         City  Salary
0    Alice   23     New York   70000
1      Bob   30  Los Angeles   80000
2  Charlie   35      Chicago  120000
3    David   40      Houston  100000
4      Eva   28      Phoenix   95000
```

[15]:
```
# Step 3: Accessing a column (e.g., 'Age')
print("\nAccessing the 'Age' column:")
print(df_from_dict['Age'])
```

```
Accessing the 'Age' column:
0    23
1    30
2    35
3    40
4    28
Name: Age, dtype: int64
```

[16]:
```
# Step 4: Accessing multiple columns (e.g., 'Name' and 'Salary')
print("\nAccessing 'Name' and 'Salary' columns:")
print(df_from_dict[['Name', 'Salary']])
```

```
Accessing 'Name' and 'Salary' columns:
      Name  Salary
0    Alice   70000
1      Bob   80000
2  Charlie  120000
3    David  100000
4      Eva   95000
```

[17]:
```
# Step 5: Filtering rows based on a condition (e.g., Age > 30)
print("\nFiltering rows where Age > 30:")
filtered_df = df_from_dict[df_from_dict['Age'] > 30]
print(filtered_df)
```

```
Filtering rows where Age > 30:
      Name  Age     City  Salary
2  Charlie   35  Chicago  120000
3    David   40  Houston  100000
```

[18]:
```
# Step 6: Adding a new column (e.g., 'Tax' calculated as 10% of Salary)
df_from_dict['Tax'] = df_from_dict['Salary'] * 0.1
print("\nDataFrame with a new 'Tax' column:")
print(df_from_dict)
```

```
DataFrame with a new 'Tax' column:
      Name  Age         City  Salary      Tax
0    Alice   23     New York   70000   7000.0
1      Bob   30  Los Angeles   80000   8000.0
2  Charlie   35      Chicago  120000  12000.0
3    David   40      Houston  100000  10000.0
4      Eva   28      Phoenix   95000   9500.0
```

[19]:
```
# Step 7: Applying a function to a column (e.g., 'Salary' increase by 5%)
df_from_dict['Salary'] = df_from_dict['Salary'] * 1.05
print("\nDataFrame with 'Salary' increased by 5%:")
print(df_from_dict)
```

```
DataFrame with 'Salary' increased by 5%:
      Name  Age         City    Salary      Tax
0    Alice   23     New York   73500.0   7000.0
1      Bob   30  Los Angeles   84000.0   8000.0
2  Charlie   35      Chicago  126000.0  12000.0
```

```
3    David   40      Houston  105000.0  10000.0
4      Eva   28      Phoenix   99750.0   9500.0
```

[20]:
```
# Step 8: Dropping a column (e.g., 'Tax')
df_from_dict = df_from_dict.drop('Tax', axis=1)
print("\nDataFrame after dropping the 'Tax' column:")
print(df_from_dict)
```

```
DataFrame after dropping the 'Tax' column:
      Name  Age         City    Salary
0    Alice   23     New York   73500.0
1      Bob   30  Los Angeles   84000.0
2  Charlie   35      Chicago  126000.0
3    David   40      Houston  105000.0
4      Eva   28      Phoenix   99750.0
```

[21]:
```
# Step 9: Handling missing data (NaN values)
df_with_nan = pd.DataFrame({
    'Product': ['Apple', 'Banana', 'Cherry', 'Date'],
    'Price': [1.2, None, 2.5, None]
})
print("\nDataFrame with missing values:")
print(df_with_nan)
```

```
DataFrame with missing values:
  Product  Price
0   Apple    1.2
1  Banana    NaN
2  Cherry    2.5
3    Date    NaN
```