Asmita Porwal
Batch-1
Day-10
1/2/2024
Data engineering

Assignment-10

Pandas for Data Processing
Reading CSV Data using Pandas

```
# Python program to illustrate
# creating a data frame using CSV files

# import pandas module
import pandas as pd

# creating a data frame
df = pd.read_csv("stu_data.csv")
print(df.head())
```

```
import pandas as pd
   Name  M1 Score  M2 Score
0  Alex        62        80
1  Brad        45        56
2  Joey        85        98
PS D:\DataEngineeringhexa\Python>
```

Read Data from CSV Files to Pandas Dataframes

```
# import pandas module
import pandas as pd
# import csv module
import csv

with open("new_data1.csv") as csv_file:
```

```
    # read the csv file
    csv_reader = csv.reader(csv_file)

    # now we can use this csv files into the pandas
    df = pd.DataFrame([csv_reader], index = None)

# iterating values of first column
for val in list(df[1]):
    print(val)
```

```
  import pandas as pd
['0', '1', 'E1', 'E6', '1642207', '1642207', '1', 'E1>E6>E2>E4>E3>E1']
PS D:\DataEngineeringhexa\Python>
```

Filter Data in Pandas Dataframe using query

```
import pandas as pd

# Create a sample DataFrame
data = {'Name': ['Alice', 'Bob', 'Charlie', 'David'],
        'Age': [25, 30, 22, 35],
        'Salary': [50000, 60000, 45000, 70000]}

df = pd.DataFrame(data)

# Use the query() method to filter data
filtered_df = df.query('Age > 25 and Salary > 50000')

# Display the filtered DataFrame
print(filtered_df)
```

```
   import pandas as pd
     Name  Age  Salary
1     Bob   30   60000
3   David   35   70000
PS D:\DataEngineeringhexa\Py
```

Get Count by Status using Pandas Dataframe APIs

```python
import pandas as pd

# Assuming you have a DataFrame named 'df' with a 'Status' column
# Create a sample DataFrame
data = {'Name': ['Alice', 'Bob', 'Charlie', 'David'],
        'Status': ['Active', 'Inactive', 'Active', 'Pending']}

df = pd.DataFrame(data)

# Use value_counts() to get the count of each unique value in 'Status'
column
status_counts = df['Status'].value_counts()

# Display the count of each status
print(status_counts)
```

```
Status
Active      2
Inactive    1
Pending     1
```

Get count by Month and Status using Pandas Dataframe APIs

```python
import pandas as pd

# Sample DataFrame
data = {'Date': ['2022-01-01', '2022-01-02', '2022-02-01', '2022-02-02',
'2022-02-03'],
        'Status': ['Active', 'Inactive', 'Active', 'Pending', 'Active']}

df = pd.DataFrame(data)

# Convert 'Date' column to datetime
df['Date'] = pd.to_datetime(df['Date'])

# Set 'Date' column as the index
df.set_index('Date', inplace=True)

print(df)
# Group by month and status, then count occurrences
result_df = df.groupby([df.index.to_period("M"),
'Status']).size().unstack(fill_value=0)

# Rename the index and columns for clarity
result_df.index = result_df.index.astype(str)
result_df.columns.name = None  # Remove the name of the columns axis

# Display the result
print(result_df)
```

```
                Status
Date
2022-01-01    Active
2022-01-02  Inactive
2022-02-01    Active
2022-02-02   Pending
2022-02-03    Active
            Active  Inactive  Pending
Date
2022-01          1         1        0
2022-02          2         0        1
```

Create Dataframes using dynamic column list on CSV Data

```python
import pandas as pd

# Specify the list of columns dynamically
columns_to_select = ['Name', 'M1 Score']

# Read CSV file using only the selected columns
file_path = 'D:\DataEngineeringhexa\Python\Stu_data.csv'
df = pd.read_csv(file_path, usecols=columns_to_select)

# Display the DataFrame
print(df)
```

```
    Name  M1 Score
0   Alex         62
1   Brad         45
2   Joey         85
PS D:\DataEngineeringhexa\Pyt
```

Performing Inner Join between Pandas Dataframes

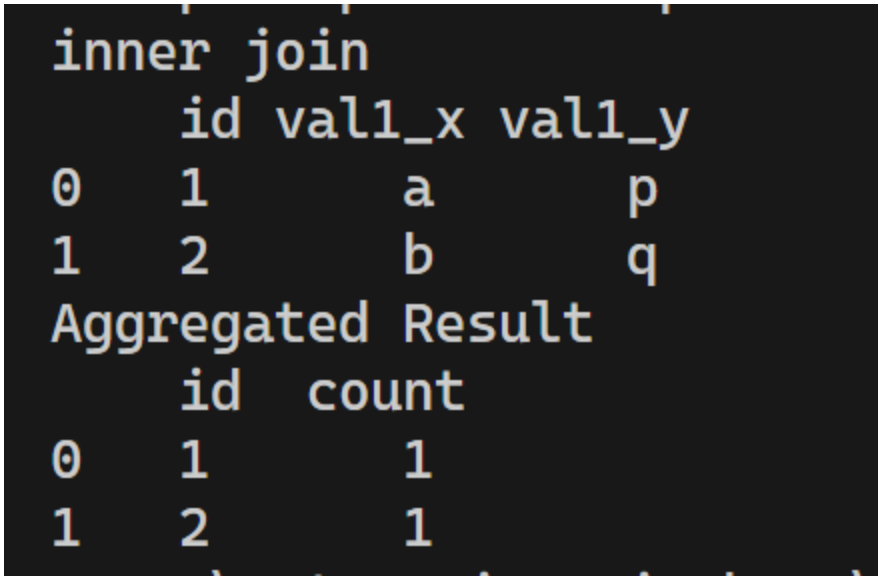
```
# importing pandas
import pandas as pd

# Creating dataframe a
a = pd.DataFrame()

# Creating Dictionary
d = {'id': [1, 2, 10, 12],
     'val1': ['a', 'b', 'c', 'd']}

a = pd.DataFrame(d)

# Creating dataframe b
b = pd.DataFrame()

# Creating dictionary
d = {'id': [1, 2, 9, 8],
     'val1': ['p', 'q', 'r', 's']}
b = pd.DataFrame(d)

# inner join
df = pd.merge(a, b, on='id', how='inner')

# display dataframe
print("inner join\n",df)
```

```
inner join
   id val1_x val1_y
0   1      a      p
1   2      b      q
PS D:\DataEngineeringhex
```

Perform Aggregations on Join results

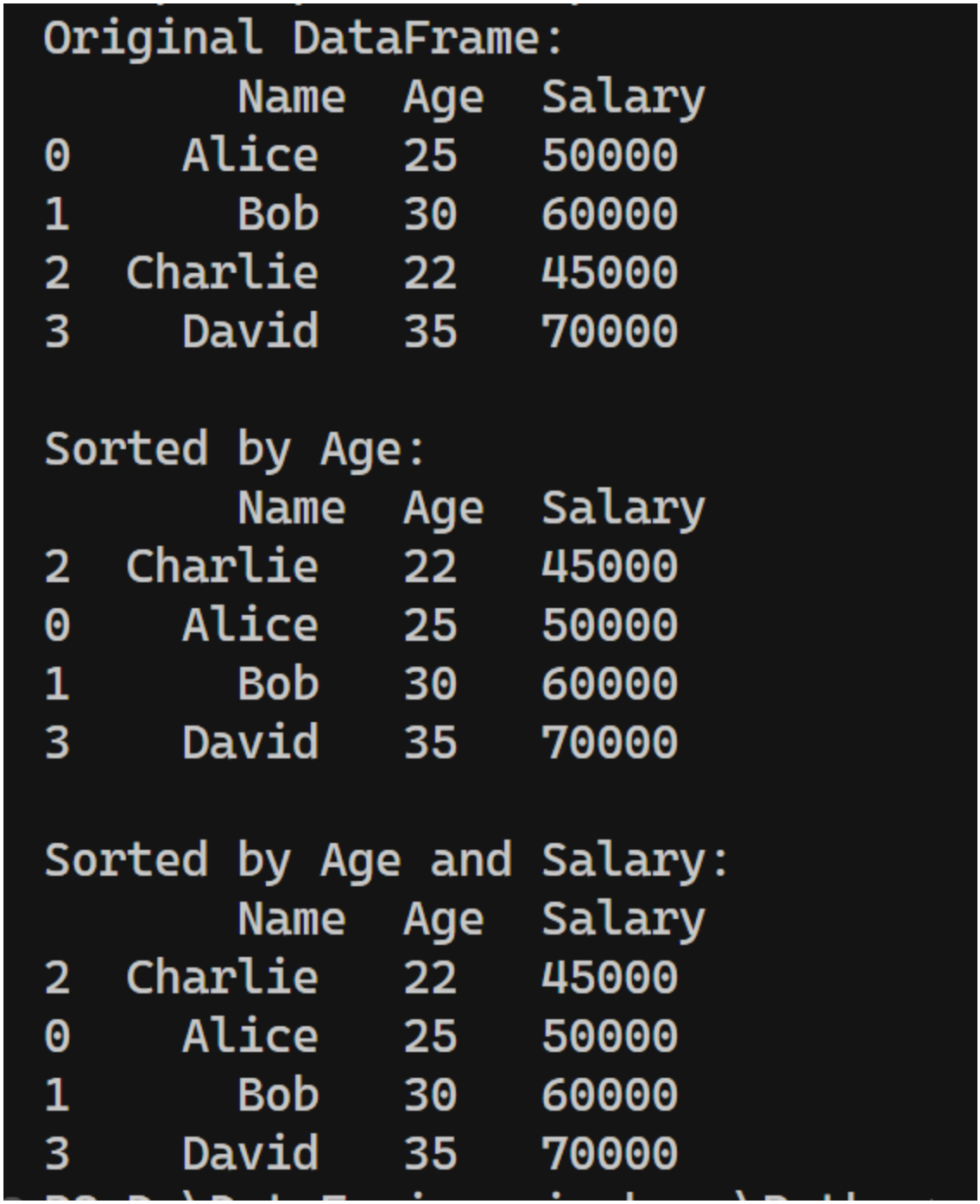
```
# importing pandas
```

```python
import pandas as pd

# Creating dataframe a
a = pd.DataFrame()

# Creating Dictionary
d = {'id': [1, 2, 10, 12],
     'val1': ['a', 'b', 'c', 'd']}

a = pd.DataFrame(d)

# Creating dataframe b
b = pd.DataFrame()

# Creating dictionary
d = {'id': [1, 2, 9, 8],
     'val1': ['p', 'q', 'r', 's']}
b = pd.DataFrame(d)

# inner join
df = pd.merge(a, b, on='id', how='inner')

# display dataframe
print("inner join\n",df)

aggregated_df = df.groupby('id').size().reset_index(name='count')
print("Aggregated Result\n", aggregated_df)
```



Sort Data in Pandas Dataframes

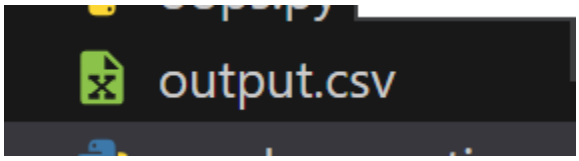
```
import pandas as pd

# Create a sample DataFrame
data = {'Name': ['Alice', 'Bob', 'Charlie', 'David'],
        'Age': [25, 30, 22, 35],
        'Salary': [50000, 60000, 45000, 70000]}

df = pd.DataFrame(data)

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

# Sort by a single column (e.g., 'Age')
df_sorted_age = df.sort_values(by='Age', ascending=True)
print("\nSorted by Age:\n", df_sorted_age)

# Sort by multiple columns (e.g., 'Age' and 'Salary')
df_sorted_multiple = df.sort_values(by=['Age', 'Salary'], ascending=[True,
False])
print("\nSorted by Age and Salary:\n", df_sorted_multiple)
```



Writing Pandas Dataframes to Files

```
import pandas as pd

# Create a sample DataFrame
data = {'Name': ['Alice', 'Bob', 'Charlie', 'David'],
```

```
        'Age': [25, 30, 22, 35],
        'Salary': [50000, 60000, 45000, 70000]}

df = pd.DataFrame(data)

# Write to CSV
df.to_csv('output.csv', index=False)
```



Write Pandas Dataframes to JSON Files

```
import pandas as pd

# Create a sample DataFrame
data = {'Name': ['Alice', 'Bob', 'Charlie', 'David'],
        'Age': [25, 30, 22, 35],
        'Salary': [50000, 60000, 45000, 70000]}

df = pd.DataFrame(data)

# Write to JSON
df.to_json('output.json', orient='records', lines=True)
```