Pandas - Data Aggregation

Basic Statistical Aggregations

```
import pandas as pd
import numpy as np

df = pd.DataFrame({
    'Name': ['Alice', 'Bob', 'Charlie', 'David'],
    'Age': [25, 30, 35, 28],
    'Score': [85, 92, 78, 88],
    'Department': ['A', 'B', 'A', 'B']
})

# Single column statistics
print("Mean age:", df['Age'].mean())
print("Max score:", df['Score'].max())
print("Median age:", df['Age'].median())
print("Std dev of scores:", df['Score'].std())

# Descriptive statistics
print("\nDescriptive statistics:")
print(df.describe())
```

GroupBy Operations

```
# Group by and aggregate
grouped = df.groupby('Department')['Score'].mean()
print("Average score by department:")
print(grouped)

# Multiple aggregations
multi_agg = df.groupby('Department').agg({
    'Score': ['mean', 'max', 'min'],
    'Age': 'mean'
})
print("\nMultiple aggregations:")
print(multi_agg)

# Count by group
counts = df.groupby('Department').size()
print("\nCount per department:")
print(counts)
```

Advanced Aggregation Functions

```
# Custom aggregation
def score_range(x):
    return x.max() - x.min()

custom = df.groupby('Department')['Score'].agg([
    'mean',
    'std',
    ('range', score_range)
])
print("Custom aggregations:")
print(custom)

# Multiple group by
df_extended = df.copy()
```

```
df_extended['Semester'] = ['Fall', 'Fall', 'Spring', 'Spring']

multi_group = df_extended.groupby(['Department', 'Semester'])['Score'].mean()
print("\nMultiple groupby:")
print(multi_group)
```

Pivot Tables

```
# Create pivot table
pivot = df.pivot_table(
    values='Score',
    index='Department',
    aggfunc=['mean', 'count']
)
print("Pivot table:")
print(pivot)

# More complex pivot
df_sales = pd.DataFrame({
    'Region': ['East', 'East', 'West', 'West'] * 2,
    'Product': ['A', 'B', 'A', 'B'] * 2,
    'Quarter': ['Q1', 'Q1', 'Q1', 'Q1', 'Q2', 'Q2', 'Q2', 'Q2'],
    'Sales': np.random.randint(100, 500, 8)
})

pivot_sales = df_sales.pivot_table(
    values='Sales',
    index='Region',
    columns='Quarter',
    aggfunc='sum',
    fill_value=0
)
print("\nSales pivot:")
print(pivot_sales)
```

Rolling and Cumulative Operations

```
# Create time series data
dates = pd.date_range('2024-01-01', periods=10, freq='D')
ts_df = pd.DataFrame({
    'Value': np.random.randint(10, 50, 10)
}, index=dates)

# Rolling mean
ts_df['Rolling_Mean_3'] = ts_df['Value'].rolling(window=3).mean()
print("Rolling mean:")
print(ts_df)

# Cumulative sum
ts_df['Cumsum'] = ts_df['Value'].cumsum()

# Expanding mean
ts_df['Expanding_Mean'] = ts_df['Value'].expanding().mean()

print("\nWith cumulative operations:")
print(ts_df)
```