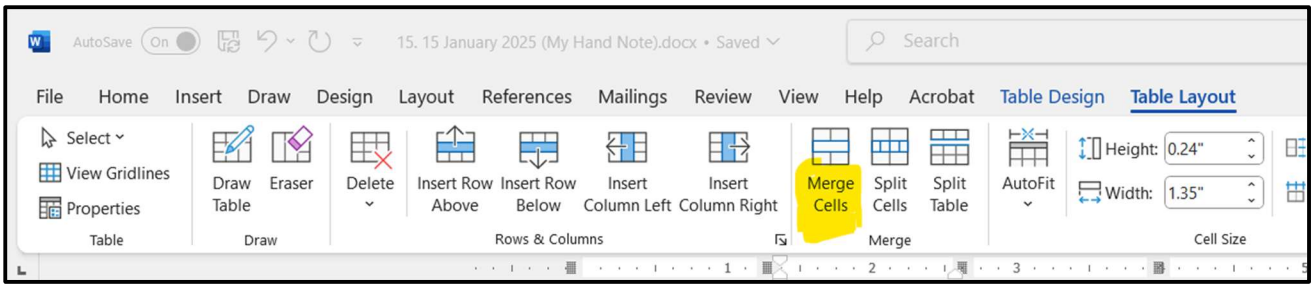
# Class #15: Working with Pandas (Infinity) - Part 2

1. Hierarchical indexing in Pandas.
2. Handling missing data in Pandas.
3. Data wrangling with Pandas.
4. Useful methods and operations in Pandas. Assignment #15:

## 1. Hierarchical indexing in Pandas.

আমরা জানি, DataFrame এবং Pandas মূলত Excel বা SQL-এর মতো টেবিল/ছক হিসেবে কাজ করে। Excel হলো একটি স্প্রেডশিট, যেখানে ডেটা সারি ও কলামে সাজানো থাকে।

Microsoft Word বা Excel-এ যখন টেবিলের সারি/কলাম মর্জ করা হয়, তখন সেই একত্রিত টেবিলটি Hierarchical indexing র মতো কাজ করে। এটি একাধিক স্তরের indexing র মাধ্যমে ডেটাকে সংগঠিত করে।




Series এ আমরা column পাবো না। সেজন্য, column value লিখা যায় না ।

pd.Series(data = my_data, index = my_labels)

একটি প্যান্ডাস Series তৈরি করে, যেখানে একটি মাল্টি-লেভেল ইনডেক্স ব্যবহার

```
import numpy as np
import pandas as pd

# Create a multi-level index
index = [['a', 'a', 'a', 'b', 'b', 'b', 'c', 'c', 'd', 'd'],  #
level 1 index
         [1, 2, 3, 1, 2, 3, 1, 2, 1, 2]]                        #
level 2 index

# Print the index using f-string
print(f"Multi-level Index:{index}")

# Generate random numbers
num = np.random.randn(10)

# Print the random numbers using f-string
```

```
print(f"\nGenerated Random Numbers:{num}")

# Create the Series with the multi-level index and random numbers
ser = pd.Series(np.random.randn(10), index=index)

# Print the Series using f-string
print(f"\nGenerated Series:\n{ser}")
```

```
Multi-level Index:[['a', 'a', 'a', 'b', 'b', 'b', 'c', 'c', 'd',
'd'], [1, 2, 3, 1, 2, 3, 1, 2, 1, 2]]

Generated Random Numbers:[ 0.90014648 -0.75532132 -0.5291819
0.04659035  0.23231563  0.0942922
 -1.64923753  0.25980859  0.70925334  0.29610676]

Generated Series:
a  1   -0.625633
   2   -0.271362
   3   -1.877020
b  1    1.580067
   2   -0.525526
   3   -1.145976
c  1    0.375690
   2   -2.081324
d  1   -0.800168
   2   -0.192844
dtype: float64
```

1. **মাল্টি-লেভেল ইনডেক্স তৈরি:** এখানে একটি তালিকা (list) ব্যবহার করা হয়েছে যা দুটি স্তরের ইনডেক্স ধারণ করে।
    - প্রথম স্তরের ইনডেক্স: ['a', 'a', 'a', 'b', 'b', 'b', 'c', 'c', 'd', 'd']
    - দ্বিতীয় স্তরের ইনডেক্স: [1, 2, 3, 1, 2, 3, 1, 2, 1, 2]

    এই দুটি স্তরের ইনডেক্স একত্রে একটি মাল্টি-লেভেল ইনডেক্স তৈরি করে।
2. **র‍্যান্ডম নম্বর জেনারেট করা:** np.random.randn(10) ব্যবহার করে ১০টি র‍্যান্ডম মান (এখানে স্বাভাবিক বণ্টনের মান) তৈরি করা হয়েছে।
3. **প্যান্ডাস সিরিজ তৈরি:** pd.Series(np.random.randn(10), index=index) ব্যবহার করে একটি সিরিজ তৈরি করা হয়েছে, যেখানে ইনডেক্স হিসেবে পূর্বের মাল্টি-লেভেল ইনডেক্স ব্যবহার করা হয়েছে এবং মান হিসেবে np.random.randn(10) দেওয়া হয়েছে।
4. **ফলাফল প্রদর্শন:** ফরম্যাটেড স্ট্রিং (f-string) ব্যবহার করে ইনডেক্স, র‍্যান্ডম নম্বর এবং সিরিজের মান কনসোল/টেক্সট আউটপুট হিসেবে প্রদর্শন করা হয়েছে।

একটি multi-level index ব্যবহার করে pd Series তৈরি করে এবং কিছু ডেটা slicing (string slicing মনে আছে ? তেমন আর কি)

```
import numpy as np
import pandas as pd
```

```
# Create a multi-level index
index = [['a', 'a', 'a', 'b', 'b', 'b', 'c', 'c', 'd', 'd'],  #
level 1 index
         [1, 2, 3, 1, 2, 3, 1, 2, 1, 2]]                        #
level 2 index
# Generate random numbers and create the Series
ser = pd.Series(np.random.randn(10), index=index)
# Print the generated Series
print(f"Generated Series:\n{ser}")

# Data retrieval for all values with 'a' in level 1 of index
print(f"\nData for index 'a':\n{ser['a']}")

# Retrieve a single value from index 'a' at level 2 with index 2
print(f"\nSingle value from index 'a' at position 2:
{ser['a'][2]}")
```

```
Generated Series:
a  1   -0.593420
   2   -0.478001
   3    0.007603
b  1    2.119662
   2   -1.587915
   3   -1.547188
c  1   -1.373043
   2    0.283126
d  1   -0.640141
   2    0.296133
dtype: float64

Data for index 'a':
1   -0.593420
2   -0.478001
3    0.007603
dtype: float64

Single value from index 'a' at position 2: -0.47800077713996464
```

ব্যাখ্যা:

1. **মাল্টি-লেভেল ইনডেক্স তৈরি:**
   - দুটি স্তরের ইনডেক্স ব্যবহার করা হয়েছে। প্রথম স্তরের ইনডেক্স (যেমন 'a', 'b', 'c', 'd') এবং দ্বিতীয় স্তরের ইনডেক্স (যেমন 1, 2, 3)।
2. **র‍্যান্ডম ডেটা তৈরি:**
   - np.random.randn(10) ব্যবহার করে ১০টি র‍্যান্ডম মান তৈরি করা হয় এবং সেই মানগুলো মাল্টি-লেভেল ইনডেক্সে দেওয়া হয়।
3. **ডেটা প্রিন্ট করা:**
   - প্রথমে পুরো সিরিজ প্রিন্ট করা হয়।
   - তারপর 'a' ইনডেক্সের সব মান দেখানো হয়।
   - শেষে, 'a' ইনডেক্সের ২ নম্বর অবস্থানে থাকা একটি নির্দিষ্ট মান প্রিন্ট করা হয়।

## Example with DataFrame:

With a DataFrame, either axis can have a hierarchical index. একটি 2D অ্যারে তৈরি করে, তারপরে সেই অ্যারে ব্যবহার করে একটি মাল্টি-লেভেল ইনডেক্স সহ প্যান্ডাস DataFrame তৈরি

```
import numpy as np
import pandas as pd

# Step 1: Create a 2D array
array_2d = np.arange(12).reshape(4, 3)

# Step 2: Define multi-level index and column labels
index_2d = [['a', 'a', 'b', 'b'], [1, 2, 1, 2]]
columns = ['AB', 'ON', 'BC']

# Step 3: Create the DataFrame
df = pd.DataFrame(array_2d, index=index_2d, columns=columns)

# Step 4: Display the results
print(f"2D Array:\n{array_2d}\n")
print(f"DataFrame:\n{df}")
```

```
2D Array:
[[ 0  1  2]
 [ 3  4  5]
 [ 6  7  8]
 [ 9 10 11]]

DataFrame:
     AB  ON  BC
a 1   0   1   2
  2   3   4   5
b 1   6   7   8
  2   9  10  11
```

DataFrame প্রদর্শন

1. **2D অ্যারে তৈরি**: np.arange(12) দিয়ে ১২টি মান তৈরি করা হয় এবং reshape(4, 3) দিয়ে সেই মানগুলিকে 4x3 আকারে রূপান্তর করা হয়।
2. **মাল্টি-লেভেল ইনডেক্স এবং কলাম লেবেল**: index দুটি স্তর ব্যবহার করা হয়েছে: প্রথম স্তর হল ['a', 'a', 'b', 'b'] এবং দ্বিতীয় স্তর হল [1, 2, 1, 2]। columns লেবেল হিসেবে ['AB', 'ON', 'BC'] ব্যবহার করা হয়েছে।
3. **DataFrame তৈরি**: pd.DataFrame() ব্যবহার করে অ্যারে, index এবং columns লেবেল দিয়ে একটি DataFrame তৈরি করা হয়।
4. **ফলাফল প্রদর্শন**: শেষে, 2D অ্যারে এবং DataFrame উভয়ই print() দিয়ে প্রদর্শন করা হয়।

How to index the above dataframe!

- on the columns axis, just use normal bracket notation df[].
- on row axis, we use df.loc[] loc iloc শুধুমাত্র dataframe এ row কাজ করে । series এ কাজ করে না ।

Calling one level of the index returns the sub-dataframe. একটি মাল্টি-লেভেল ইনডেক্স সহ DataFrame তৈরি করে এবং তার পর বিভিন্ন ভাবে ডেটা অ্যাক্সেস করে. (Slicing like String)

```
import numpy as np
import pandas as pd

# Step 1: Create a 2D array
array_2d = np.arange(12).reshape(4, 3)

# Step 2: Define multi-level index and column labels
index_2d = [['a', 'a', 'b', 'b'], [1, 2, 1, 2]]
columns = ['AB', 'ON', 'BC']

# Step 3: Create the DataFrame
df = pd.DataFrame(array_2d, index=index_2d, columns=columns)

# Step 4: Display the DataFrame
print(f"DataFrame:\n{df}\n")

# Step 5: Print the 'AB' column
print(f"'AB' Column:\n{df['AB']}\n")

# Step 6: Print rows with index 'b'
print(f"Rows with index 'b':\n{df.loc['b']}\n")

# Step 7: Print value at index 'b' and level 2 index 2
print(f"Value at index 'b' and level 2 index 2:
\n{df.loc['b'].loc[2]}\n")

# Step 8: Print value at index 'b', level 2 index 2, column 'BC'
print(f"Value at index 'b', level 2 index 2, column 'BC':
{df.loc['b'].loc[2]['BC']}")
```

```
DataFrame:
     AB  ON  BC
a 1   0   1   2
  2   3   4   5
b 1   6   7   8
  2   9  10  11

'AB' Column:
a  1    0
   2    3
b  1    6
   2    9
Name: AB, dtype: int32

Rows with index 'b':
   AB  ON  BC
```

```
1   6   7   8
2   9  10  11

Value at index 'b' and level 2 index 2:
AB     9
ON    10
BC    11
Name: 2, dtype: int32

Value at index 'b', level 2 index 2, column 'BC': 11
```

1. **2D অ্যারে তৈরি**: np.arange(12) দিয়ে ১২টি মান তৈরি করা হয় এবং reshape(4, 3) দিয়ে সেই মানগুলিকে 4x3 আকারে রূপান্তর করা হয়।
2. **মাল্টি-লেভেল ইনডেক্স এবং কলাম লেবেল**: index দুটি স্তর ব্যবহার করা হয়েছে: প্রথম স্তর হল ['a', 'a', 'b', 'b'] এবং দ্বিতীয় স্তর হল [1, 2, 1, 2]। columns লেবেল হিসেবে ['AB', 'ON', 'BC'] ব্যবহার করা হয়েছে।
3. **DataFrame তৈরি**: pd.DataFrame() ব্যবহার করে অ্যারে, index এবং columns লেবেল দিয়ে একটি DataFrame তৈরি করা হয়।

**Step 5: 'AB' Column**

- এখানে, 'AB' কলামটি আলাদাভাবে প্রদর্শন করা হচ্ছে। এটি df['AB'] দ্বারা করা হচ্ছে, যেখানে 'AB' কলামে থাকা সমস্ত ডেটা দেখতে পাচ্ছি। এই কলামের সব মানগুলো একত্রে দেখানো হয়।

**Step 6: Rows with index 'b'**

- df.loc['b'] দিয়ে b ইনডেক্সের সাথে সম্পর্কিত সব রেকর্ড বের করা হয়। অর্থাৎ, প্রথম স্তরের ইনডেক্স b হওয়া সমস্ত সারির তথ্য দেখতে পাওয়া যায়, যেখানে দ্বিতীয় স্তরের ইনডেক্সও ১ ও ২ হবে।

**Step 7: Value at index 'b' and level 2 index 2**

- df.loc['b'].loc[2] দিয়ে প্রথমে b ইনডেক্সের সব ডেটা নেওয়া হয়, তারপর .loc[2] ব্যবহার করে দ্বিতীয় স্তরের ইনডেক্স 2 এর তথ্য বের করা হয়। এইভাবে, আমরা b ইনডেক্সের দ্বিতীয় স্তরের ২ নম্বর রেকর্ডটি দেখতে পাই।

**Step 8: Value at index 'b', level 2 index 2, column 'BC'**

- এখানে, df.loc['b'].loc[2]['BC'] দিয়ে প্রথমে b ইনডেক্সের ডেটা বের করা হয়, তারপর .loc[2] দিয়ে দ্বিতীয় স্তরের ইনডেক্স ২ এর ডেটা বের করা হয়, এবং শেষে 'BC' কলামের নির্দিষ্ট মান (এখানে ৮) দেখা হয়। এটি একটি নির্দিষ্ট মান, যেমন 8 ফেরত দেয়।

loc হচ্ছে location এর সংক্ষিপ্তরুপ. row তে dataframe থাকলে df.loc[] ব্যবহার করতে হয় ।

একটি 2D NumPy অ্যারে তৈরি করে এবং সেটি Pandas DataFrame এ রূপান্তরিত করে, যার মধ্যে মাল্টি-লেভেল ইনডেক্স রয়েছে

```
import numpy as np
import pandas as pd

# Create a 2D array and reshape it
array_2d = np.arange(12).reshape(4, 3)

# Define the index and column labels
```

```
index_2d = [['a', 'a', 'b', 'b'], [1, 2, 1, 2]]
columns = ['AB', 'ON', 'BC']

# Create the DataFrame
df = pd.DataFrame(array_2d, index=index_2d, columns=columns)

# Display the DataFrame
print(f"DataFrame:\n{df}\n")

# Show index names before naming them
print(f"Index names before naming: {df.index.names}")

# Assign names to the index levels
df.index.names = ['Level_1', 'Level_2']

# Display the DataFrame with named index levels
print(f"\nDataFrame with named index levels:\n{df}")
```

1. **2D অ্যারে তৈরি:** np.arange(12) দিয়ে ১২টি মান তৈরি করা হয় এবং reshape(4, 3) দিয়ে সেই মানগুলিকে 4x3 আকারে রূপান্তর করা হয়।
2. **মাল্টি-লেভেল ইনডেক্স এবং কলাম লেবেল:** index দুটি স্তর ব্যবহার করা হয়েছে: প্রথম স্তর হল ['a', 'a', 'b', 'b'] এবং দ্বিতীয় স্তর হল [1, 2, 1, 2]। columns লেবেল হিসেবে ['AB', 'ON', 'BC'] ব্যবহার করা হয়েছে।
3. **DataFrame তৈরি:** pd.DataFrame() ব্যবহার করে অ্যারে, index এবং columns লেবেল দিয়ে একটি DataFrame তৈরি করা হয়।
4. **ইনডেক্সের নাম দেখা:** df.index.names ব্যবহার করে ইনডেক্সের নাম দেখা হয় (যা আগে None থাকবে)।
5. **ইনডেক্সের নাম নির্ধারণ করা:** df.index.names = ['Level_1', 'Level_2'] দিয়ে ইনডেক্সের স্তরের নাম দেওয়া হয়।
6. **নামযুক্ত ইনডেক্স সহ DataFrame দেখানো:** নাম সহ DataFrame আবার কনসোলে প্রদর্শন করা হয়।

## 3. Data wrangling with Pandas.

এটি ডেটা প্রক্রিয়া করার একটি প্রক্রিয়া, যাতে কাঁচা ডেটাকে পরিষ্কার এবং বিশ্লেষণের জন্য প্রস্তুত করা হয়। প্যান্ডাস ব্যবহার করে ডেটার মধ্যে কোনও ত্রুটি (missing values, duplicate rows, wrong formats ইত্যাদি) থাকলে তা ঠিক করা, ডেটা রূপান্তর (data transformation), গ্রুপিং, এবং ফিল্টারিং করা হয়।

**উদাহরণ:**

- Missing values পূর্ণ করা (fillna())
- ডেটা ফিল্টার করা (filtering data)
- ডেটা গ্রুপিং (groupby())
- ডেটা কম্বাইন করা (merge(), concat())

## Groupby:

Groupby pandas একটি গুরুত্বপূর্ণ ফাংশনালিটি যা ডেটাকে গ্রুপ করে, অ্যাগ্রিগেট ফাংশন প্রয়োগ করে এবং ফলাফলগুলো একত্রিত করে। এর তিনটি ধাপ:

1. **Split:** ডেটাকে গ্রুপে ভাগ করা হয় নির্দিষ্ট কী-এর ভিত্তিতে।
2. **Apply:** প্রতিটি গ্রুপে একটি ফাংশন প্রয়োগ করা হয়।
3. **Combine:** ফলাফলগুলো একত্রিত করে নতুন একটি অবজেক্ট তৈরি করা হয়।

ডেটাফ্রেম তৈরির পর সেটিতে নতুন একটি কলাম যোগ করে তাতে মান দেয়ার পদ্ধতি:

```
import numpy as np
import pandas as pd

# Step 1: Create the dataframe
data = {
    'Store': ['Walmart', 'Walmart', 'Costco', 'Costco', 'Target',
'Target'],
    'Customer': ['Tim', 'Jermy', 'Mark', 'Denice', 'Ray', 'Sam'],
    'Sales': [150, 200, 550, 90, 430, 120]
}
df = pd.DataFrame(data)

# Step 2: Print the original dataframe
print(f"Dataframe:\n{df}")

# Step 3: Add a new column 'qty' with value 10 for each row
df['qty'] = 10

# Step 4: Print the updated dataframe with 'qty' column
print(f"\nUpdated Dataframe with 'qty' column:\n{df}")
```

```
Dataframe:
     Store Customer  Sales
0  Walmart      Tim    150
1  Walmart    Jermy    200
2   Costco     Mark    550
3   Costco   Denice     90
4   Target      Ray    430
5   Target      Sam    120

Updated Dataframe with 'qty' column:
     Store Customer  Sales  qty
0  Walmart      Tim    150   10
1  Walmart    Jermy    200   10
2   Costco     Mark    550   10
3   Costco   Denice     90   10
4   Target      Ray    430   10
5   Target      Sam    120   10
```

1. **DataFrame তৈরি:** প্রথমে একটি ডেটাফ্রেম/table তৈরি করা হয়েছে, যেখানে ৩টি কলাম আছে: 'Store', 'Customer', এবং 'Sales'।

2. **মৌলিক DataFrame প্রিন্ট:** তৈরি করা ডেটাফ্রেমটি পর্দায় প্রদর্শন করা হয়েছে। custome index না দেয়ায় 0,1,2 এভাবে দেখিয়েছে।
3. **নতুন কলাম যোগ:** নতুন একটি কলাম 'qty' তৈরি করা হয়েছে, এবং তার প্রতিটি রো-তে মান ১০ দেওয়া হয়েছে।
4. **আপডেটেড DataFrame প্রিন্ট:** আপডেট করা ডেটাফ্রেমটি আবার পর্দায় প্রদর্শন করা হয়েছে, যাতে নতুন 'qty' কলামটি দেখা যায়।

ডেটাফ্রেম তৈরি করা থেকে শুরু করে, গ্রুপিং এবং যোগফল বের করার কাজ করা:

```
import pandas as pd

# Step 1: Create the dataframe
data = {
    'Store': ['Walmart', 'Walmart', 'Costco', 'Costco', 'Target',
'Target'],
    'Customer': ['Tim', 'Jermy', 'Mark', 'Denice', 'Ray', 'Sam'],
    'Sales': [150, 200, 550, 90, 430, 120]
}
df = pd.DataFrame(data)

# Step 2: Add a new column 'qty' with value 10 for all rows
df['qty'] = 10

# Step 3: Print the updated dataframe
print(f"\nUpdated DataFrame:\n{df}")

# Step 4: Group by 'Store' and sum the 'Sales'
sales_by_store = df.groupby("Store")['Sales'].sum()
print(f"\nSales Summed by Store:\n{sales_by_store}")

# Step 5: Group by 'Store' and sum all columns
by_store = df.groupby("Store")
print(f"\nTotal Sum by Store (including 'Sales' and
'qty'):\n{by_store.sum()}")

# Step 6: Get the sum of all columns for 'Target' store in one
line
target_sum = df.groupby('Store').sum().loc['Target']
print(f"\nTotal Sum for 'Target' store:\n{target_sum}")
```

```
Updated DataFrame:
     Store Customer  Sales  qty
0  Walmart      Tim    150   10
1  Walmart    Jermy    200   10
2   Costco     Mark    550   10
3   Costco   Denice     90   10
4   Target      Ray    430   10
5   Target      Sam    120   10

Sales Summed by Store:
```

```
Store
Costco      640
Target      550
Walmart     350
Name: Sales, dtype: int64

Total Sum by Store (including 'Sales' and 'qty'):
           Customer  Sales  qty
Store
Costco   MarkDenice    640   20
Target       RaySam    550   20
Walmart    TimJermy    350   20

Total Sum for 'Target' store:
Customer    RaySam
Sales          550
qty             20
Name: Target, dtype: object
```

1. **ডেটাফ্রেম তৈরি**: প্রথমে একটি ডেটাফ্রেম তৈরি করা হয়েছে, যেখানে তিনটি কলাম রয়েছে — 'Store', 'Customer', এবং 'Sales'।
2. **নতুন কলাম যোগ করা**: ডেটাফ্রেমে একটি নতুন কলাম 'qty' যোগ করা হয়েছে, যার মান সব রো-তে ১০।
3. **ডেটাফ্রেম প্রিন্ট করা**: আপডেট হওয়া ডেটাফ্রেমটি প্রিন্ট করা হয়েছে, যাতে 'qty' কলামটি দেখা যায়।
4. **'Store' অনুযায়ী 'Sales' এর মোট যোগফল**: 'Store' অনুযায়ী গ্রুপ করে, প্রতিটি স্টোরের জন্য 'Sales' এর মোট যোগফল বের করা হয়েছে এবং প্রিন্ট করা হয়েছে।
5. **সব কলামের মোট যোগফল**: 'Store' অনুযায়ী আবার গ্রুপ করে, সব কলাম (যেমন 'Sales' এবং 'qty') এর মোট যোগফল বের করা হয়েছে এবং প্রিন্ট করা হয়েছে।
6. **'Target' স্টোরের জন্য সব কলামের মোট যোগফল**: এক লাইনে 'Target' স্টোরের জন্য সব কলামের যোগফল বের করা হয়েছে এবং প্রিন্ট করা হয়েছে।

```
import pandas as pd

# Step 1: Create the dataframe
data = {
    'Store': ['Walmart', 'Walmart', 'Costco', 'Costco', 'Target',
'Target'],
    'Customer': ['Tim', 'Jermy', 'Mark', 'Denice', 'Ray', 'Sam'],
    'Sales': [150, 200, 550, 90, 430, 120]
}
df = pd.DataFrame(data)
# Step 2: Add a new column 'qty' with value 10 for all rows
df['qty'] = 10
# Step 3: Print the updated dataframe
print(f"\nUpdated DataFrame:\n{df}")

# Step 4: Group by 'Store' and perform operations on columns
by_store = df.groupby("Store")
```

```
# Get the minimum values in each group (by_store)
print(f"\nMinimum values by Store:\n{by_store.min()}")

# Get the maximum values in each group (by_store)
print(f"\nMaximum values by Store:\n{by_store.max()}")

# Count the number of occurrences of each column in each group
(by_store)
# This works with strings as well
print(f"\nCount of values by Store:\n{by_store.count()}")
```

```
Updated DataFrame:
     Store Customer  Sales  qty
0  Walmart      Tim    150   10
1  Walmart    Jermy    200   10
2   Costco     Mark    550   10
3   Costco   Denice     90   10
4   Target      Ray    430   10
5   Target      Sam    120   10

Minimum values by Store:
         Customer  Sales  qty
Store
Costco    Denice     90   10
Target       Ray    120   10
Walmart    Jermy    150   10

Maximum values by Store:
         Customer  Sales  qty
Store
Costco       Mark    550   10
Target        Sam    430   10
Walmart       Tim    200   10

Count of values by Store:
          Customer  Sales  qty
Store
Costco           2      2    2
Target           2      2    2
Walmart          2      2    2
```

**1. ডেটাফ্রেম তৈরি:**

প্রথমে একটি ডেটাফ্রেম তৈরি করা হয়েছে, যেখানে তিনটি কলাম রয়েছে:

- **'Store'**: স্টোরের নাম (যেমন Walmart, Costco, Target)
- **'Customer'**: গ্রাহকের নাম
- **'Sales'**: বিক্রয়ের পরিমাণ

এই ডেটা একটি pandas ডেটাফ্রেমে পরিণত করা হয়েছে।

**2. নতুন কলাম 'qty' যোগ করা:** ডেটাফ্রেমে একটি নতুন কলাম qty যোগ করা হয়েছে, যার মান সব রো-তে ১০।

**3. ডেটাফ্রেম প্রিন্ট করা:**

আপডেট করা ডেটাফ্রেমটি প্রিন্ট করা হয়েছে, যাতে নতুন qty কলাম সহ পুরো ডেটা দেখা যায়।

**4. গ্রুপিং এবং অপারেশন:**

ডেটাফ্রেমটি **'Store'** কলামের ভিত্তিতে গ্রুপ করা হয়েছে, তারপর প্রতিটি গ্রুপের উপর কিছু অপারেশন করা হয়েছে:

- **Minimum values:** প্রতিটি স্টোরের জন্য সর্বনিম্ন মান বের করা হয়েছে (যেমন, 'Sales' কলামের সর্বনিম্ন মান)।
- **Maximum values:** প্রতিটি স্টোরের জন্য সর্বোচ্চ মান বের করা হয়েছে (যেমন, 'Sales' কলামের সর্বোচ্চ মান)।

['Tim', 'Jermy', 'Mark', 'Denice', 'Ray', 'Sam'] এদের জন্য ইংরেজি অক্ষরের ক্রম হিসাব করা হয়েছে। max and min.

- **Count:** প্রতিটি স্টোরে কতটি রো আছে, তা গোনা হয়েছে (যেমন, 'Customer' এবং 'Sales' কলামের সংখ্যা)।

**আউটপুট:** প্রতিটি স্টোরের জন্য সর্বনিম্ন, সর্বোচ্চ এবং সংখ্যা বের করা হবে এবং সেগুলো প্রিন্ট করা হবে। এভাবে প্রোগ্রামটি ডেটাফ্রেমে গ্রুপিং করে বিভিন্ন ধরনের পরিসংখ্যান বের করার জন্য কাজ করেছে।