Final Review Session

May 8, 2024

0.0.1 Final Review Session

[3]:
```
from sklearn.preprocessing import LabelBinarizer
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

Handling Missing Values

[4]:
```
data = {
    'Employee_ID': [101, 102, 103, 104, 105, 106, 107, 108, 109, 110],
    'Name': ['Ahmed', 'Bob', 'Charlie', 'David', 'Eve', 'Corey', 'Hussain',␣
  ↪'Jacob', 'Motaz', 'Elmer'],
    'Age': [25, np.nan, 30, 35, 40, 22, 43, 32, 21, 65],
    'Department': ['HR', 'IT', np.nan, 'Finance', 'Operations', 'Finance',␣
  ↪'Operations', 'HR', 'IT', 'IT'],
    'Salary': [50000, 60000, 70000, np.nan, 80000, 50000, 60000, 17000, 20000,␣
  ↪40000]
}

emp_df = pd.DataFrame(data)
emp_df
```

[4]:
```
   Employee_ID     Name   Age  Department   Salary
0          101    Ahmed  25.0          HR  50000.0
1          102      Bob   NaN          IT  60000.0
2          103  Charlie  30.0         NaN  70000.0
3          104    David  35.0     Finance      NaN
4          105      Eve  40.0  Operations  80000.0
5          106    Corey  22.0     Finance  50000.0
6          107  Hussain  43.0  Operations  60000.0
7          108    Jacob  32.0          HR  17000.0
8          109    Motaz  21.0          IT  20000.0
9          110    Elmer  65.0          IT  40000.0
```

[40]:
```
emp_df[np.sum(emp_df.isna(), axis=1)==0]
```

```
[40]:    Employee_ID     Name   Age  Department   Salary
      0          101    Ahmed  25.0          HR  50000.0
      4          105      Eve  40.0  Operations  80000.0
      5          106    Corey  22.0     Finance  50000.0
      6          107  Hussain  43.0  Operations  60000.0
      7          108    Jacob  32.0          HR  17000.0
      8          109    Motaz  21.0          IT  20000.0
      9          110    Elmer  65.0          IT  40000.0
```

Axis 0 (column wise) vs Axis 1 (row wise)

```
[32]: np.sum(emp_df.isna(), axis=0)
```

```
[32]: Employee_ID    0
      Name           0
      Age            1
      Department     1
      Salary         1
      dtype: int64
```

```
[41]: np.sum(emp_df.isna(), axis=1)
```

```
[41]: 0    0
      1    1
      2    1
      3    1
      4    0
      5    0
      6    0
      7    0
      8    0
      9    0
      dtype: int64
```

Group By with Aggregation

```
[6]: emp_df[['Age','Department','Salary']].groupby('Department').agg({'Age':'mean', 'Salary':('min', 'max')})
```

```
[6]:              Age   Salary
                 mean      min      max
     Department
     Finance     28.5  50000.0  50000.0
     HR          28.5  17000.0  50000.0
     IT          43.0  20000.0  60000.0
     Operations  41.5  60000.0  80000.0
```

Example: Library Data

[7]:

```
data = {
    'ID': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
    'Floor': [1, 1, 2, 2, 3, 3, 4, 4, 5, 5],
    'Section Name': ['Fiction', 'Non-Fiction', 'Science', 'Mathematics',␣
  ↪'History',
                     'Biography', 'Poetry', 'Art', 'Children', 'Reference'],
    'Section Head': ['John Doe', 'John Doe', 'Jane Smith', 'Jane Smith', 'Mark␣
  ↪Johnson',
                     'Mark Johnson', 'Emily Brown', 'Emily Brown', 'Sarah␣
  ↪Wilson', 'Sarah Wilson'],
    'No. of Books Available': [100, 80, 120, 90, 110, 70, 95, 75, 85, 100],
    'No. of Books Lent': [20, 15, 25, 10, 30, 5, 15, 10, 10, 20]
}

lib_df = pd.DataFrame(data)
lib_df
```

[7]:

```
   ID  Floor Section Name  Section Head  No. of Books Available  \
0   1      1      Fiction      John Doe                     100
1   2      1  Non-Fiction      John Doe                      80
2   3      2      Science    Jane Smith                     120
3   4      2  Mathematics    Jane Smith                      90
4   5      3      History  Mark Johnson                     110
5   6      3    Biography  Mark Johnson                      70
6   7      4       Poetry   Emily Brown                      95
7   8      4          Art   Emily Brown                      75
8   9      5     Children  Sarah Wilson                      85
9  10      5    Reference  Sarah Wilson                     100

   No. of Books Lent
0                 20
1                 15
2                 25
3                 10
4                 30
5                  5
6                 15
7                 10
8                 10
9                 20
```

[8]:

```
lib_df[['Section Name', 'Section Head', 'No. of Books Available','No. of Books␣
  ↪Lent']].groupby(['Section Head', 'Section Name']).sum()
```

[8]:

```
                           No. of Books Available  No. of Books Lent
Section Head Section Name
```

```
Emily Brown  Art                              75                  10
             Poetry                           95                  15
Jane Smith   Mathematics                      90                  10
             Science                         120                  25
John Doe     Fiction                         100                  20
             Non-Fiction                      80                  15
Mark Johnson Biography                        70                   5
             History                         110                  30
Sarah Wilson Children                         85                  10
             Reference                       100                  20
```

Data Filtering

[26]:
```
(lib_df['No. of Books Lent']>= 20) & (lib_df['No. of Books Available']< 115)
```

[26]:
```
0     True
1    False
2    False
3    False
4     True
5    False
6    False
7    False
8    False
9     True
dtype: bool
```

[27]:
```
lib_df[(lib_df['No. of Books Lent']>= 20) & (lib_df['No. of Books Available']<␣
 ↪115)]
```

[27]:
```
   ID  Floor Section Name  Section Head  No. of Books Available  \
0   1      1      Fiction      John Doe                     100
4   5      3      History  Mark Johnson                     110
9  10      5    Reference  Sarah Wilson                     100

   No. of Books Lent
0                 20
4                 30
9                 20
```

Label Binarization

[11]:
```
data = ['cat', 'dog', 'bird', 'cat', 'dog']
lb = LabelBinarizer()
binary_data = lb.fit_transform(data)
print(binary_data)
```

```
[[0 1 0]
```

```
 [0 0 1]
 [1 0 0]
 [0 1 0]
 [0 0 1]]
```

[14]:

```
data = {
    'ID': [1, 2, 3, 4, 5],
    'Name': ['Chair', 'Table', 'Sofa', 'Bed', 'Desk'],
    'Type': ['Wooden', 'Metal', 'Plastic', 'Wooden', 'Metal']
}

# Convert data into DataFrame
df = pd.DataFrame(data)

# Perform label binarization on the 'Type' column
lb = LabelBinarizer()
binary_type = lb.fit_transform(df['Type'])

# Convert binary_type into DataFrame
binary_type_df = pd.DataFrame(binary_type, columns=lb.classes_)

# Concatenate the binary_type_df with the original DataFrame
df = pd.concat([df, binary_type_df], axis=1)

df
```

[14]:

```
   ID   Name     Type  Metal  Plastic  Wooden
0   1  Chair   Wooden      0        0       1
1   2  Table    Metal      1        0       0
2   3   Sofa  Plastic      0        1       0
3   4    Bed   Wooden      0        0       1
4   5   Desk    Metal      1        0       0
```

Binning

[12]:

```
import pandas as pd

data = {'Age': [22, 35, 47, 55, 68, 72, 28, 32, 45, 51]}

df = pd.DataFrame(data)
df
```

[12]:

```
   Age
0   22
1   35
2   47
3   55
4   68
```

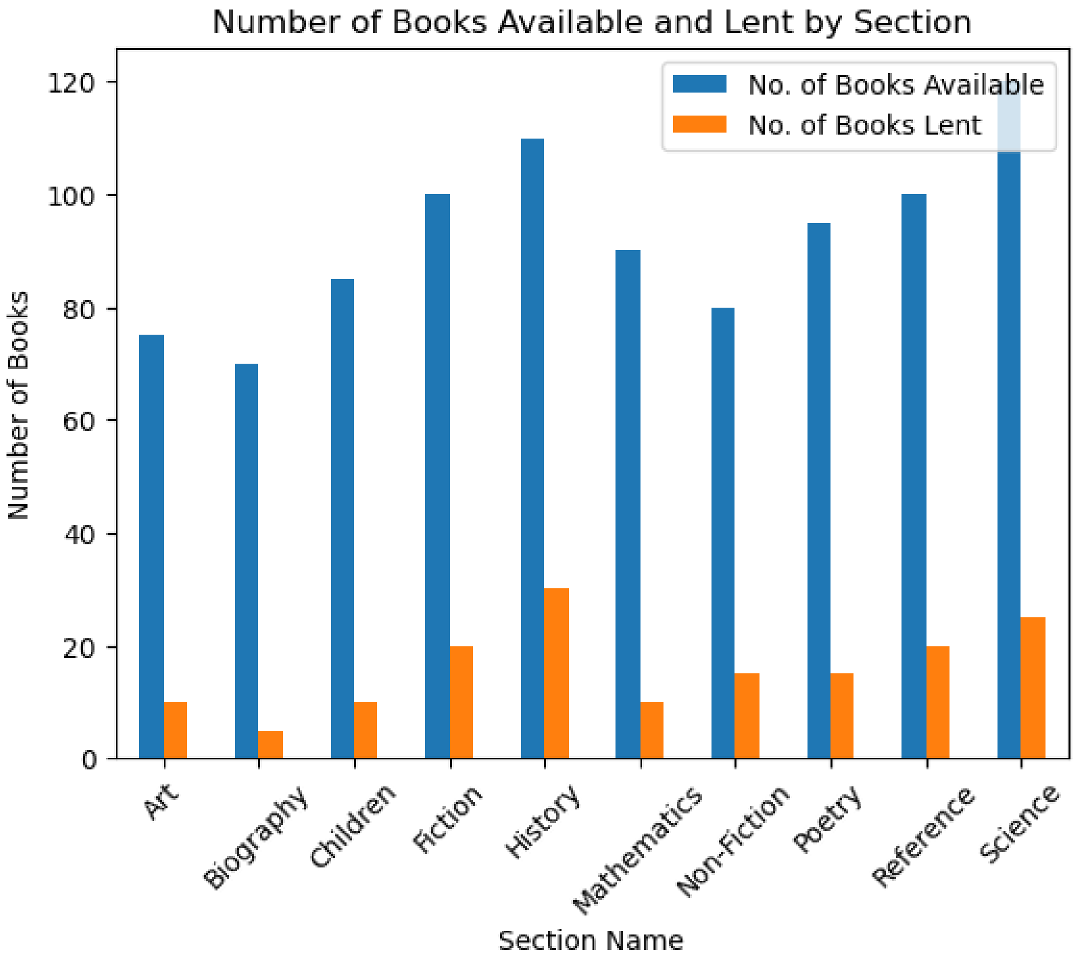
```
5   72
6   28
7   32
8   45
9   51
```

[13]:
```
# Define bin edges
bins = [0, 30, 50, 100]

labels = ['Young', 'Middle-aged', 'Senior']

df['Age Group'] = pd.cut(df['Age'], bins=bins, labels=labels)

df
```

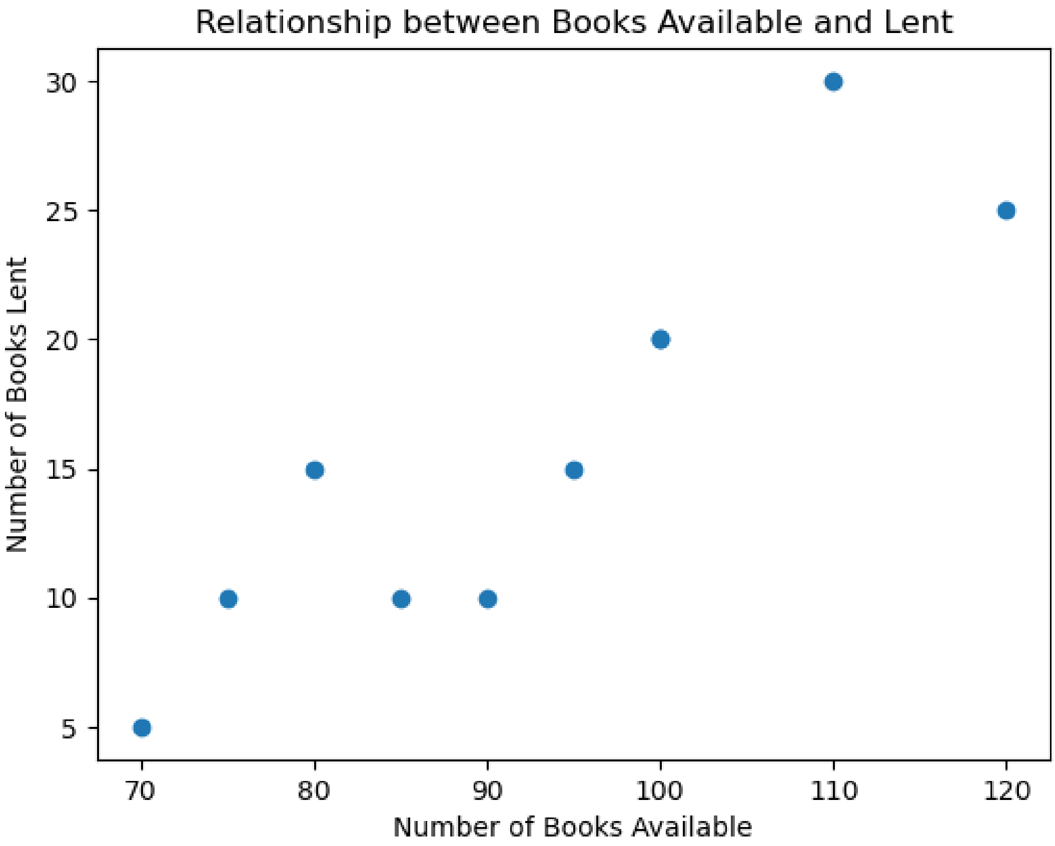
[13]:
```
   Age    Age Group
0   22        Young
1   35  Middle-aged
2   47  Middle-aged
3   55       Senior
4   68       Senior
5   72       Senior
6   28        Young
7   32  Middle-aged
8   45  Middle-aged
9   51       Senior
```

Data Visualization Examples

[15]:
```
import matplotlib.pyplot as plt

# Group the data by Section Name and calculate the sum of books available and␣
 ↪lent
section_stats = lib_df.groupby('Section Name').sum()

# Plot bar chart
section_stats.plot(kind='bar', y=['No. of Books Available', 'No. of Books␣
 ↪Lent'],
                   xlabel='Section Name', ylabel='Number of Books',
                   title='Number of Books Available and Lent by Section')
plt.xticks(rotation=45)
plt.show()
```



```
[16]: # Plot scatter plot
plt.scatter(lib_df['No. of Books Available'], lib_df['No. of Books Lent'])
plt.xlabel('Number of Books Available')
plt.ylabel('Number of Books Lent')
plt.title('Relationship between Books Available and Lent')
plt.show()
```



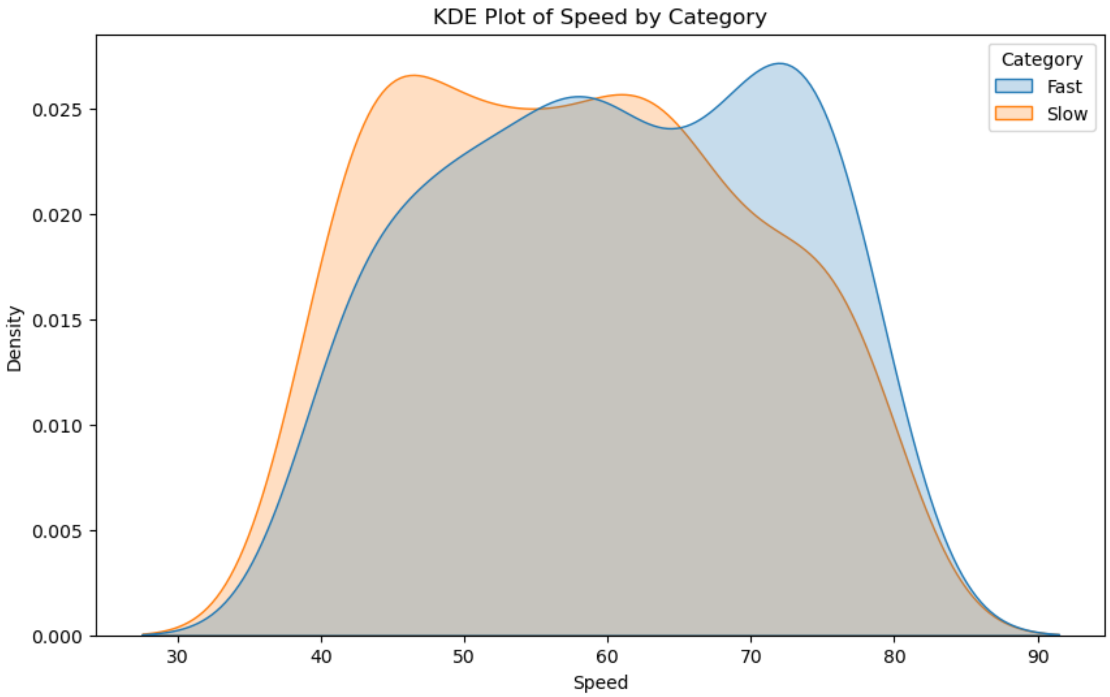
```
[18]: np.random.seed(0)
      np.random.seed(0)
      time = pd.date_range(start='2024-01-01', periods=350, freq='H')
      speed = np.random.randint(40, 80, size=350)
      category = np.random.choice(['Fast', 'Slow'], size=350)

      traffic_data = pd.DataFrame({'Time': time, 'Speed': speed, 'Category':␣
        ↪category})
      traffic_data.head()
```

```
/tmp/ipykernel_2066/2314165996.py:3: FutureWarning: 'H' is deprecated and will
be removed in a future version, please use 'h' instead.
  time = pd.date_range(start='2024-01-01', periods=350, freq='H')
```

```
[18]:                  Time  Speed Category
      0 2024-01-01 00:00:00     40     Fast
      1 2024-01-01 01:00:00     43     Fast
      2 2024-01-01 02:00:00     43     Slow
      3 2024-01-01 03:00:00     79     Slow
      4 2024-01-01 04:00:00     49     Fast
```

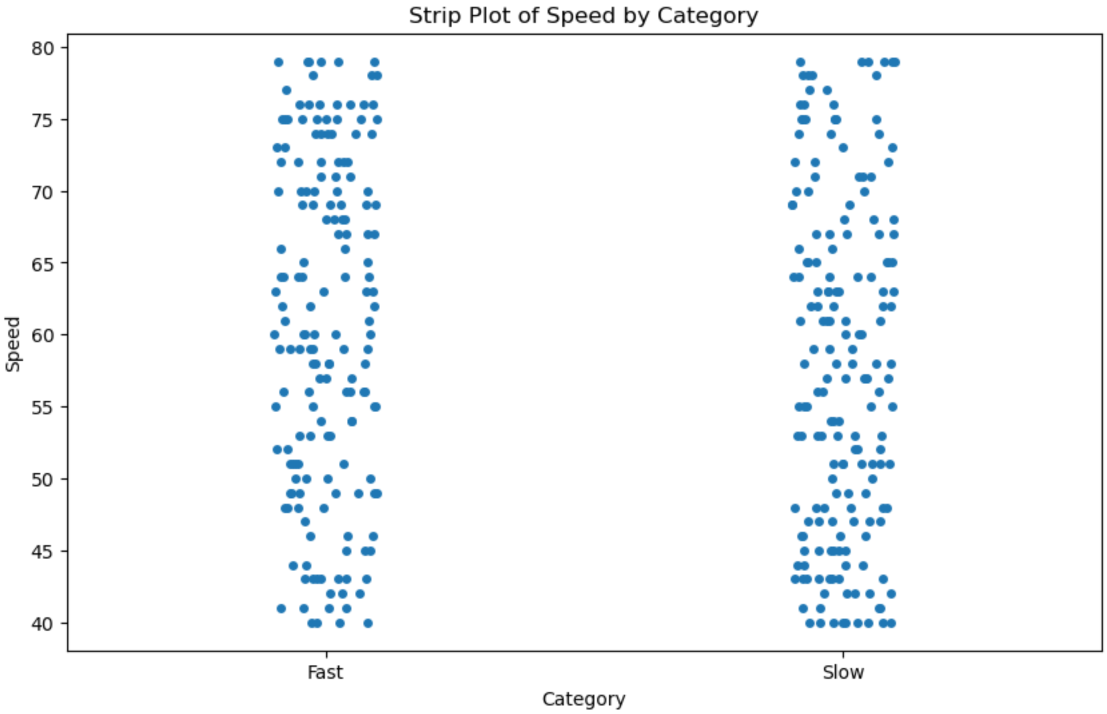
[19]:

```
plt.figure(figsize=(10, 6))
sns.kdeplot(data=traffic_data, x='Speed', hue='Category', fill=True,␣
 ↪common_norm=False)
plt.title('KDE Plot of Speed by Category')
plt.xlabel('Speed')
plt.ylabel('Density')
plt.show()
```



[20]:

```
plt.figure(figsize=(10, 6))
sns.stripplot(data=traffic_data, x='Category', y='Speed', jitter=True)
plt.title('Strip Plot of Speed by Category')
plt.xlabel('Category')
plt.ylabel('Speed')
plt.show()
```



Web Scraping

[21]:
```
import requests
from bs4 import BeautifulSoup

url = 'https://www.geeksforgeeks.org/python-programming-language/'
response = requests.get(url)
soup = BeautifulSoup(response.text, 'html.parser')
x = soup.find_all('h2')
for z in x:
    print(z.text)
```

```
What is Python?
Writing your first Python Program to Learn Python Programming
Python3
Table of Content
Setting up Python
Getting Started with Python Programming
Learn Python Input/Output
Python Data Types
Python Operators
Python Conditional Statement
Python Loops
Python Functions
Python OOPs Concepts
Python Exception Handling
Python Packages or Libraries
Python Collections
Python Database Handling
Python vs. Other Programming Languages
Learn More About Python with Different Applications:
Python Online Quiz
Python Latest & Upcoming Features
What kind of Experience do you want to share?
```

[22]:

```
url = 'https://www.geeksforgeeks.org/python-programming-language/'
response = requests.get(url)
soup = BeautifulSoup(response.text, 'html.parser')
x = soup.find_all('a')
for z in x[-5:]:
    print(z.text.strip())
```

```
Work Experiences
Campus Experiences
Competitive Exam Experiences
Can't choose a topic to write? click here for suggested topics
Write and publish your own Article
```

Pivot vs. Group By

[33]:

```
# importing pandas
import pandas as pd

# creating dataframe
df = pd.DataFrame({'Product': ['Carrots', 'Broccoli', 'Banana', 'Banana',
                                                        'Beans', 'Orange',␣
 ↪'Broccoli', 'Banana'],
                                'Category': ['Vegetable', 'Vegetable', 'Fruit',␣
 ↪'Fruit',
                                                                'Vegetable',␣
 ↪'Fruit', 'Vegetable', 'Fruit'],
                                'Quantity': [8, 5, 3, 4, 5, 9, 11, 8],
                                'Amount': [270, 239, 617, 384, 626, 610, 62,␣
 ↪90]})
df
```

[33]:

```
    Product   Category  Quantity  Amount
0   Carrots  Vegetable         8     270
1  Broccoli  Vegetable         5     239
2    Banana      Fruit         3     617
3    Banana      Fruit         4     384
4     Beans  Vegetable         5     626
```

```
5     Orange      Fruit         9     610
6   Broccoli  Vegetable        11      62
7     Banana      Fruit         8      90
```

[35]:

```
pivot = df.pivot_table(index=['Product'],
                                        values=['Amount'],
                                        aggfunc='sum')
pivot
```

[35]:

```
          Amount
Product
Banana      1091
Beans        626
Broccoli     301
Carrots      270
Orange       610
```

[37]:

```
df[['Product', 'Amount']].groupby('Product').sum()
```

[37]:

```
          Amount
Product
Banana      1091
Beans        626
Broccoli     301
Carrots      270
Orange       610
```

Difference between Pivot and Group By:

- Pivot is primarily used for reshaping or restructuring data, rotating rows into columns or vice versa whereas Group by is primarily used for aggregation, summarizing data based on one or more key columns.
- Pivot produces a new table with reshaped data, often resulting in a multi-level index or hierarchical columns whereas Group by produces a summary or aggregation of data, typically in a Series or DataFrame with aggregated values.
- Pivot allows you to specify columns to use as index, columns, and values whereas Group by allows you to specify one or more key columns for grouping and apply aggregation functions on one or more columns.

[ ]: