**17-03-2025 Training Day – 30**

# Topic: Styling Matplotlib Visualizations

* Experimented with styles like seaborn-dark and ggplot.
* Example: Customized a bar chart with labels, gridlines, and colors. visualization easier and identifying outliers easily.

Matplotlib comes with a variety of built-in styles that can be applied to your plots with a single line of code. These styles can dramatically change the look and feel of your plots, making them more suitable for different purposes like presentations, reports, or technical papers

from matplotlib import style

1. IQR: It stand for "inter quartile range", which define as the difference of "third quartile(q3) and first quartile (q0)".
2. Outliers are those value which comes after the last quartile to affect our mean, as well as below the first quartile.
3. Our whole data is divided in four part i.e. 25%, 50%, 75%, 100%, and these percentile values refers to our quartile(q1,q2,q3,q4).

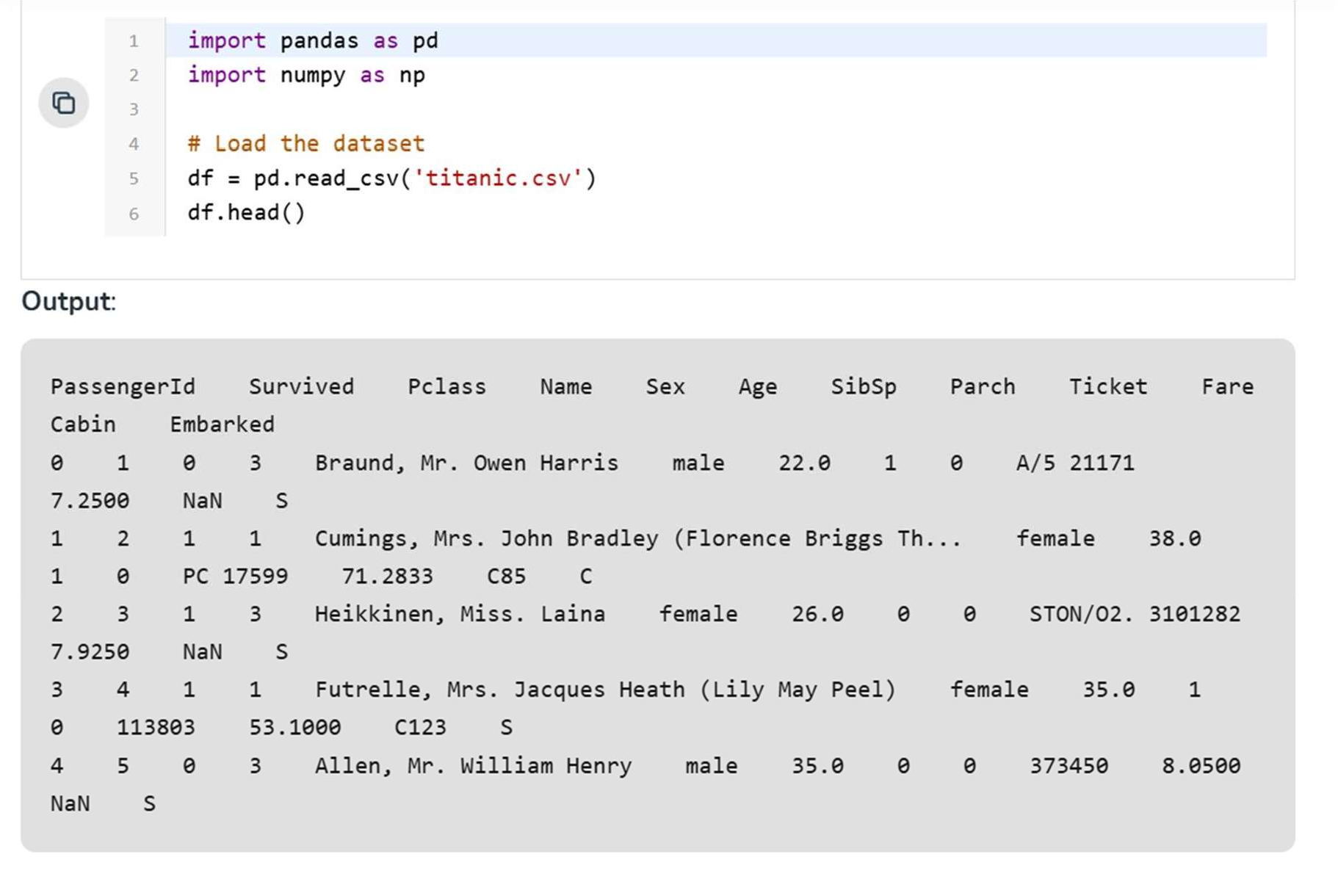


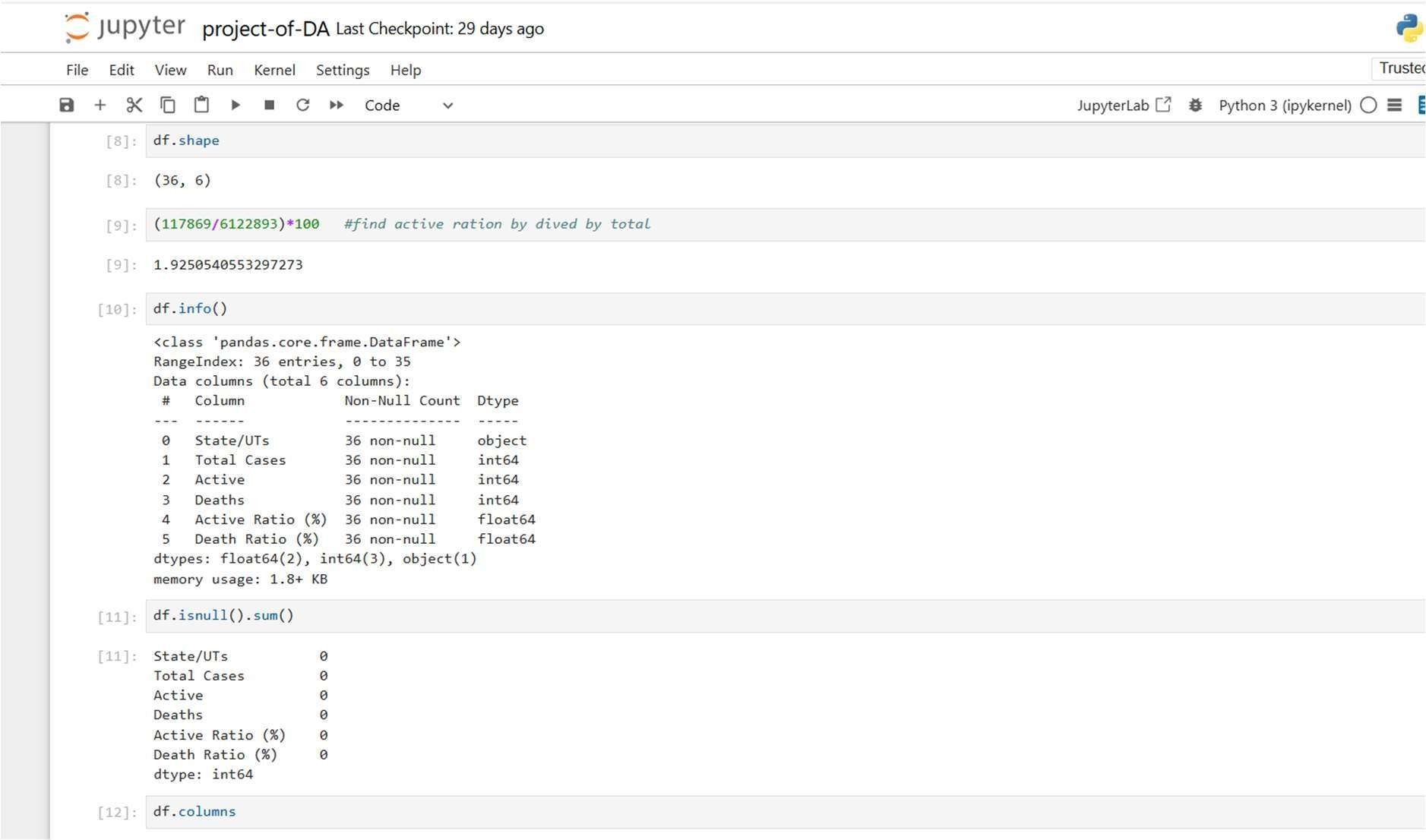
**18-03-2025 Training Day – 31**

# Topic: Data Cleaning

* + Handled missing values and duplicates in a dataset.
  + Example: Used fillna() to replace missing values with the column mean. visualization easier and identifying outliers easily.

What is data cleaning? Data cleaning is the process of fixing or removing incorrect, corrupted, incorrectly formatted, duplicate, or incomplete data within a dataset. When combining multiple data sources, there are many opportunities for data to be duplicated or mislabeled





**19-03-2025 Training Day – 32**

# Topic: Standardizing Data

* + Applied transformations to ensure consistency in data formatting.
  + Example: Converted all text columns to uppercase using .str.upper**().**

Data standardization is an important technique that is mostly performed as a pre-processing step before inputting data into many machine learning models, to standardize the range of features of an input data set.

# Standardizing Data in Excel

Excel STANDARDIZE is available under Excel Statistical Functions. It returns a normalized value, which is also called Z-score.

The mean and standard deviation are the basis of the z-score. The z-score (or standard score) is a method to standardize scores across the same scale. It divides a score's deviation by the standard deviation in a data set. The resulting score is the standard deviation of a data point from the mean.

Zero is the average of all z-scores for a dataset. A negative z score indicates that the value is lower than the mean. A positive z score indicates that the value is higher than the mean.

**Z-Score Formula** = STANDARDIZE(x, mean, standard\_dev)

**Here:** X= data value that you need to normalize. **Mean**= Distribution arithmetic mean Standard\_dev= Distribution standard deviation.

**20-03-2025 Training Day – 32**

# Topic: Combining Datasets with Pandas

* + Learned to concatenate and merge datasets.
  + Example: Merged two datasets on a common key using pd.merge().set.

**Combining Multiple Datasets:**

#### Concatenation:

Combine along rows or columns.

python Copy code

df1 = pd.DataFrame({'A': [1, 2]})

df2 = pd.DataFrame({'A': [3, 4]}) combined = pd.concat([df1, df2])

#### Merging (Join):

Combine based on a common key.

python Copy code

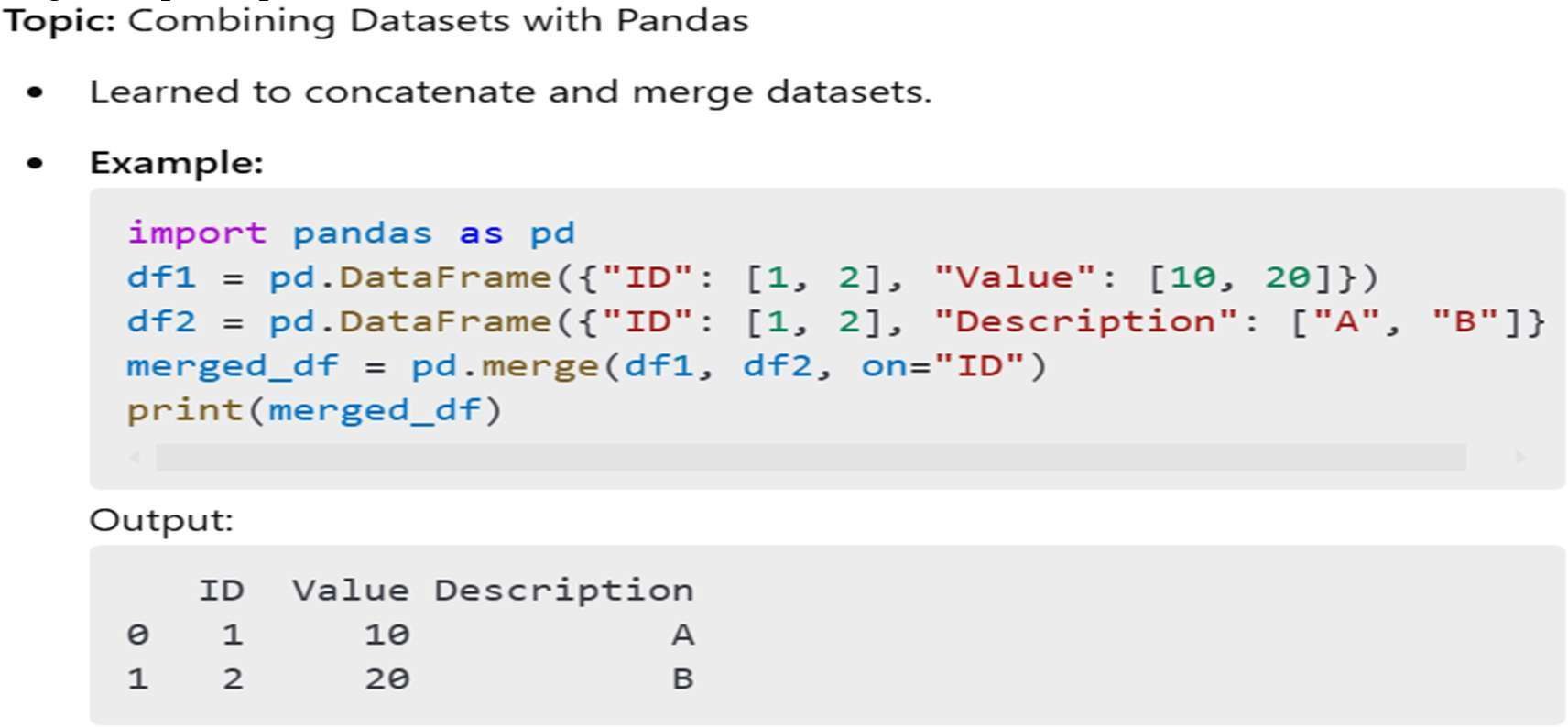
df1 = pd.DataFrame({'ID': [1, 2], 'Name': ['Alice', 'Bob']})

df2 = pd.DataFrame({'ID': [1, 2], 'Score': [85, 90]}) merged = pd.merge(df1, df2, on='ID')

#### Cleaning After Merging:

Ensure no duplicate or irrelevant columns remain.

Copy code merged.dropna(inplace=True)



**21-03-2025 Training Day – 33**

## November 12, Tuesday\*

**Topic: Advanced Groupby Operations**

## Applied multiple aggregation functions to grouped data.

* Example: Calculated mean and max for grouped columns.

### Advanced Groupby Operations

***Import Libraries***

python Copy code

import pandas as pd import numpy as np ***Create the Dataset*** python

Copy code

data = {

"Department": ["HR", "HR", "IT", "IT", "Finance", "Finance", "HR", "IT"],

"Employee": ["Alice", "Bob", "Charlie", "David", "Eve", "Frank", "Grace", "Hank"], "Salary": [50000, 60000, 80000, 90000, 70000, 75000, 62000, 88000],

"Bonus": [5000, 7000, 10000, 12000, 8000, 8500, 6000, 11000],

"Years": [2, 3, 5, 6, 4, 4, 3, 7]

}

df = pd.DataFrame(data) df

*Output of Dataset*

**Department Employee Salary Bonus Years**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| HR | Alice | 50000 | 5000 | 2 |
| HR | Bob | 60000 | 7000 | 3 |
| IT | Charlie | 80000 | 10000 | 5 |
| IT | David | 90000 | 12000 | 6 |
| Finance | Eve | 70000 | 8000 | 4 |
| Finance | Frank | 75000 | 8500 | 4 |
| HR | Grace | 62000 | 6000 | 3 |
| IT | Hank | 88000 | 11000 | 7 |

### Applying Advanced Groupby Operations

1. *Multiple Aggregations*

python Copy code

grouped = df.groupby("Department").agg({ "Salary": ["mean", "sum", "max"],

"Bonus": ["sum", "max"], "Years": ["mean"]

})

print(grouped)

*Output*

**Salary Bonus Years**

Department mean sum max sum max mean HR 57333.33 172000 62000 18000 7000 2.67

|  |  |  |
| --- | --- | --- |
| IT | 86000.00 258000 90000 33000 12000 | 6.00 |
| Finance | 72500.00 145000 75000 16500 8500 | 4.00 |

1. *Custom Aggregation Function*

python Copy code

def custom\_salary\_range(series):

return series.max() - series.min()

grouped\_custom = df.groupby("Department").agg({ "Salary": ["mean", custom\_salary\_range], "Bonus": "sum"

})

print(grouped\_custom)

*Output*

**Salary Bonus**

Department mean custom\_salary\_range sum

|  |  |  |
| --- | --- | --- |
| HR | 57333.33 12000 | 18000 |
| IT | 86000.00 10000 | 33000 |
| Finance | 72500.00 5000 | 16500 |

1. *Broadcasting Aggregation Results*

python Copy code

df["Total Salary by Dept"] = df.groupby("Department")["Salary"].transform("sum") df["Max Bonus by Dept"] = df.groupby("Department")["Bonus"].transform("max") print(df)

*Output*

**Department Employee Salary Bonus Years Total Salary by Dept Max Bonus by Dept**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| HR | Alice | 50000 | 5000 | 2 | 172000 | 7000 |
| HR | Bob | 60000 | 7000 | 3 | 172000 | 7000 |
| IT | Charlie | 80000 | 10000 | 5 | 258000 | 12000 |
| IT | David | 90000 | 12000 | 6 | 258000 | 12000 |
| Finance | Eve | 70000 | 8000 | 4 | 145000 | 8500 |
| Finance | Frank | 75000 | 8500 | 4 | 145000 | 8500 |
| HR | Grace | 62000 | 6000 | 3 | 172000 | 7000 |
| IT | Hank | 88000 | 11000 | 7 | 258000 | 12000 |
| **Discussion** |  |  |  |  |  |  |

Advanced groupby operations are crucial for deriving insights from grouped data. These techniques include:

* + Applying multiple aggregation functions.
  + Using custom functions to extract specific insights.
  + Broadcasting results back to the original dataset for further analysis.