Indexing, filtering, and aggregation example:

Sure, here's an example of indexing, filtering, and aggregation in pandas:

```
import pandas as pd

# create a DataFrame
data = {'name': ['Alice', 'Bob', 'Charlie', 'David'],
        'age': [25, 30, 35, 40],
        'city': ['New York', 'Paris', 'London', 'San Francisco']}
df = pd.DataFrame(data)

# select a specific column
ages = df['age']
print(ages)

# filter rows based on a condition
young_people = df[df['age'] < 35]
print(young_people)

# group the data by a column and calculate the mean age for each group
mean_age_by_city = df.groupby('city')['age'].mean()
print(mean_age_by_city)
```

Output:

```
0    25
1    30
2    35
3    40
Name: age, dtype: int64
      name  age       city
0    Alice   25   New York
1      Bob   30      Paris
3    David   40  San Francisco
              age
city
London     35
New York   25
Paris      30
San Francisco  40
```

In this example, we first created a DataFrame using a dictionary of data, as in the previous example. We then demonstrated how to perform indexing, filtering, and aggregation on the DataFrame.

To select a specific column, we used square bracket notation to extract the 'age' column and assigned it to a new variable `ages`.

To filter rows based on a condition, we used a boolean expression to create a boolean mask that we used to filter the DataFrame. The resulting DataFrame `young_people` contains only rows where the age is less than 35.

Finally, we used the `groupby()` method to group the data by the 'city' column, and calculated the mean age for each group using the `mean()` method. The resulting DataFrame `mean_age_by_city` contains two columns: 'city' and 'age', where the 'age' column contains the mean age for each city.