Loading data from files or databases example:

Sure, here's an example of loading data from a CSV file and a SQL database in pandas:

```
import pandas as pd

# load data from a CSV file
df_csv = pd.read_csv('data.csv')

# load data from a SQL database
# assume a database 'mydb' exists with table 'mytable'
import sqlite3
conn = sqlite3.connect('mydb.db')
df_sql = pd.read_sql_query('SELECT * FROM mytable', conn)
conn.close()
```

In this example, we first loaded data from a CSV file named 'data.csv' using the `read_csv()` function. This function reads the file and returns a DataFrame with the data.

We then demonstrated how to load data from a SQL database using the `read_sql_query()` function. First, we created a connection to the database using the `sqlite3.connect()` function, specifying the name of the database file. We then used the `read_sql_query()` function to execute a SQL query and return the results as a DataFrame. Finally, we closed the database connection using the `close()` method.

Note that to use `read_sql_query()` with other databases like MySQL or PostgreSQL, you will need to provide the appropriate database connection string and driver information.

Cleaning and transforming data example:

Sure, here's an example of cleaning and transforming data in pandas:

```
import pandas as pd

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

# clean and transform the data
df['age'] = pd.to_numeric(df['age'], errors='coerce')  # convert 'age' column
to numeric, ignoring errors
df['city'] = df['city'].str.title()  # capitalize the first letter of each
word in 'city' column
df.dropna(inplace=True)  # drop rows with missing data
```

```
# display the resulting DataFrame
print(df)
```

Output:

```
markdown
      name   age           city
0    Alice  25.0       New York
2  Charlie  35.0         London
```

In this example, we first created a DataFrame with some messy data. The 'age' column contains a mixture of integer and string values, and the 'city' column contains inconsistent capitalization.

To clean and transform the data, we first used the `to_numeric()` function to convert the 'age' column to numeric values, ignoring errors. This function returns a new series with the converted values. We assigned this new series back to the 'age' column in the original DataFrame.

Next, we used the `str.title()` method to capitalize the first letter of each word in the 'city' column. This method returns a new series with the capitalized values. We assigned this new series back to the 'city' column in the original DataFrame.

Finally, we used the `dropna()` method to drop any rows with missing data. This method returns a new DataFrame with the rows removed. We used the `inplace=True` argument to modify the original DataFrame in place.

The resulting DataFrame has only two rows with valid data, where the 'age' column contains numeric values and the 'city' column has consistent capitalization.

Calculating summary statistics example:

Sure, here's an example of calculating summary statistics in pandas:

```
import pandas as pd

# create a DataFrame with sample data
data = {'name': ['Alice', 'Bob', 'Charlie', 'David', 'Emma'],
        'age': [25, 30, 35, 40, 45],
        'salary': [50000, 60000, 70000, 80000, 90000]}
df = pd.DataFrame(data)

# calculate summary statistics
mean_age = df['age'].mean()  # mean of 'age' column
max_salary = df['salary'].max()  # maximum of 'salary' column
min_salary = df['salary'].min()  # minimum of 'salary' column
median_salary = df['salary'].median()  # median of 'salary' column
std_dev_age = df['age'].std()  # standard deviation of 'age' column
```

```
# display the summary statistics
print(f"Mean age: {mean_age}")
print(f"Maximum salary: {max_salary}")
print(f"Minimum salary: {min_salary}")
print(f"Median salary: {median_salary}")
print(f"Standard deviation of age: {std_dev_age}")
```

Output:

```
Mean age: 35.0
Maximum salary: 90000
Minimum salary: 50000
Median salary: 70000.0
Standard deviation of age: 7.905694150420948
```

In this example, we first created a DataFrame with sample data consisting of 'name', 'age', and 'salary' columns.

We then used various DataFrame methods to calculate summary statistics on the data. For example, we used the `mean()` method to calculate the mean of the 'age' column, the `max()` method to find the maximum of the 'salary' column, the `min()` method to find the minimum of the 'salary' column, and the `median()` method to find the median of the 'salary' column. We assigned these summary statistics to variables for later display.

Finally, we used the `std()` method to calculate the standard deviation of the 'age' column. This method returns a scalar value, which we assigned to a variable for later display.

The resulting output shows the calculated summary statistics.