Q1. Write a code to print the data present in the second row of the dataframe, df.

Assuming you have a pandas DataFrame named df, you can print the data present in the second row using the iloc function as follows:

In [1]:
```
import pandas as pd

# Assuming df is your DataFrame
# Example DataFrame
data = {'Name': ['John', 'Alice', 'Bob'],
        'Age': [25, 30, 35],
        'City': ['New York', 'London', 'Paris']}

df = pd.DataFrame(data)

# Print the data in the second row
second_row_data = df.iloc[1]
print(second_row_data)
```

```
Name     Alice
Age         30
City    London
Name: 1, dtype: object
```

Q2. What is the difference between the functions loc and iloc in pandas.DataFrame?

Difference between loc and iloc in pandas.DataFrame:

loc (Label-based indexing):

It is used for selection by label. It includes the last element of the range specified. The syntax is df.loc[row_indexer, column_indexer]. Example: df.loc[1, 'Age'] selects the element in the second row and 'Age' column. iloc (Integer-location based indexing):

It is used for selection by position. It excludes the last element of the range specified. The syntax is df.iloc[row_indexer, column_indexer]. Example: df.iloc[1, 1] selects the element in the second row and second column. In summary, loc uses labels to index data, while iloc uses integer positions. Choose the appropriate one based on whether you want to use labels or positions for indexing.

Q3. Reindex the given dataframe using a variable, reindex = [3,0,1,2] and store it in the variable, new_df then find the output for both new_df.loc[2] and new_df.iloc[2]. Did you observe any difference in both the outputs? If so then explain it. Consider the below code to answer further questions: import pandas as pd import numpy as np columns = ['column_1', 'column_2', 'column_3', 'column_4', 'column_5', 'column_6'] indices = [1,2,3,4,5,6]

Creating a dataframe:

df1 = pd.DataFrame(np.random.rand(6,6), columns = columns, index = indices)

In [3]:
```
import pandas as pd
import numpy as np
columns = ['column_1', 'column_2', 'column_3', 'column_4', 'column_5', 'column_6
indices = [1,2,3,4,5,6]
#Creating a dataframe:
df1 = pd.DataFrame(np.random.rand(6,6), columns = columns, index = indices)
```

Q4. Write a code to find the following statistical measurements for the above dataframe df1: (i) mean of each and every column present in the dataframe. (ii) standard deviation of column, 'column_2'

In [4]:
```
# (i) Mean of each column
mean_of_columns = df1.mean()

# (ii) Standard deviation of column 'column_2'
std_dev_column_2 = df1['column_2'].std()

print("Mean of each column:")
print(mean_of_columns)
print("\nStandard deviation of column 'column_2':", std_dev_column_2)
```

```
Mean of each column:
column_1    0.382943
column_2    0.555540
column_3    0.489532
column_4    0.690872
column_5    0.619669
column_6    0.283541
dtype: float64

Standard deviation of column 'column_2': 0.33165545116387884
```

Q5. Replace the data present in the second row of column, 'column_2' by a string variable then find the mean of column, column_2. If you are getting errors in executing it then explain why. [Hint: To replace the data use df1.loc[] and equate this to string data of your choice.]

In [5]:
```
# Replace the data in the second row of 'column_2' with a string variable
df1.loc[2, 'column_2'] = 'Replacement String'

# Find the mean of 'column_2'
# This will result in an error because 'column_2' contains non-numeric data afte
# df1['column_2'].mean()
```

Q6. What do you understand about the windows function in pandas and list the types of windows functions?

In pandas, a window function is an operation that performs a calculation across a set of rows related to the current row. It is often used for time series data. Types of window functions include:

1. Rolling: Provides a rolling view of the data.

2.Expanding: Calculates the expanding window mean. 3.EWMA (Exponential Weighted Moving Average): Provides exponential decay. 4.Aggregate: Aggregates values within a window.

Q7. Write a code to print only the current month and year at the time of answering this question. [Hint: Use pandas.datetime function]

In [6]:
```
import pandas as pd

# Print current month and year
current_date = pd.to_datetime('today')
print("Current Month and Year:", current_date.strftime('%B %Y'))
```

```
Current Month and Year: March 2024
```

Q9. Write a Python program that reads a CSV file containing categorical data and converts a specified column to a categorical data type. The program should prompt the user to enter the file path, column name, and category order, and then display the sorted data.

Python Program for Converting a Column to Categorical Data

In [ ]:
```
import pandas as pd

def convert_column_to_categorical(file_path, column_name, category_order):
    # Read CSV file into a DataFrame
    df = pd.read_csv(file_path)

    # Convert the specified column to categorical with the given order
    df[column_name] = pd.Categorical(df[column_name], categories=category_order,

    # Display sorted data
    sorted_data = df.sort_values(by=[column_name])
    print(sorted_data)

# Example usage:
file_path = input("Enter the file path of the CSV file: ")
column_name = input("Enter the column name to convert to categorical: ")
category_order_str = input("Enter the category order (comma-separated values): "
category_order = [cat.strip() for cat in category_order_str.split(',')]
convert_column_to_categorical(file_path, column_name, category_order)
```

Q10. Write a Python program that reads a CSV file containing sales data for different products and visualizes the data using a stacked bar chart to show the sales of each product category over time. The program should prompt the user to enter the file path and display the chart.

Python Program for Visualizing Sales Data Using Stacked Bar Chart

In [ ]:
```
import pandas as pd
import matplotlib.pyplot as plt

def visualize_sales_data(file_path):
    # Read CSV file into a DataFrame
    df = pd.read_csv(file_path)

    # Pivot the data for stacked bar chart
    df_pivot = df.pivot(index='Time', columns='Product Category', values='Sales'

    # Plot stacked bar chart
    df_pivot.plot(kind='bar', stacked=True)
    plt.title('Sales Data by Product Category Over Time')
    plt.xlabel('Time')
    plt.ylabel('Sales')
    plt.show()

# Example usage:
file_path = input("Enter the file path of the CSV file: ")
visualize_sales_data(file_path)
```

Q11. You are given a CSV file containing student data that includes the student ID and their test score. Write a Python program that reads the CSV file, calculates the mean, median, and mode of the test scores, and displays the results in a table. The program should do the followingM I Prompt the user to enter the file path of the CSV file containing the student dataR I Read the CSV file into a Pandas DataFrameR I Calculate the mean, median, and mode of the test scores using Pandas toolsR I Display the mean, median, and mode in a table. Assume the CSV file contains the following columnsM I Student ID: The ID of the studentR I Test Score: The score of the student's test. Example usage of the program: Enter the file path of the CSV file containing the student data: student_data.csv +-----------+--------+ | Statistic | Value | +-----------+--------+ | Mean | 79.6 | | Median | 82 | | Mode | 85, 90 | +-----------+--------+ Assume that the CSV file student_data.csv contains the following data: Student ID,Test Score 1,85 2,90 3,80 4,75 5,85 6,82 7,78 8,85 9,90 10,85 The program should calculate the mean, median, and mode of the test scores and display the results in a table.

In [ ]:
```
import pandas as pd

def analyze_student_test_scores(file_path):
    # Read CSV file into a DataFrame
    df = pd.read_csv(file_path)

    # Calculate mean, median, and mode of test scores
    mean_score = df['Test Score'].mean()
    median_score = df['Test Score'].median()
    mode_scores = df['Test Score'].mode()

    # Display results in a table
    results_table = pd.DataFrame({'Statistic': ['Mean', 'Median', 'Mode'],
                                  'Value': [mean_score, median_score, ', '.join(
    print(results_table)

# Example usage:
```

```
file_path = input("Enter the file path of the CSV file: ")
analyze_student_test_scores(file_path)
```

In [ ]: