assignment=21

July 28, 2023

[7]:
```
#Consider following code to answer further questions:
import pandas as pd
course_name = ['Data Science', 'Machine Learning', 'Big Data', 'Data Engineer']
duration = [2,3,6,4]
df = pd.DataFrame(data = {'course_name' : course_name, 'duration' : duration})
```

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

[8]:
```
df.loc[1]
```

[8]:
```
course_name    Machine Learning
duration                      3
Name: 1, dtype: object
```

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

loc and iloc are both methods in the pandas library that allow you to select data from a DataFrame based on indices or labels.

loc is primarily label-based, meaning that it is used to select data based on row labels and column names. You can use loc to select rows and columns by passing the row label(s) and column label(s) as arguments. For example, to select the value in row 'A' and column 'B' of a DataFrame df, you would use df.loc['A', 'B'].

iloc, on the other hand, is primarily integer-based, meaning that it is used to select data based on the integer position of the rows and columns. You can use iloc to select rows and columns by passing the row index(es) and column index(es) as arguments. For example, to select the value in the first row and second column of a DataFrame df, you would use df.iloc[0, 1].

3 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:

[9]:
```
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)
```

[10]:
```
new_df = df1.reindex([3,0,1,2])
```

[11]:
```
new_df.loc[2]
```

[11]:
```
column_1    0.743944
column_2    0.636201
column_3    0.129964
column_4    0.933156
column_5    0.104330
column_6    0.573110
Name: 2, dtype: float64
```

[12]:
```
new_df.iloc[2]
```

[12]:
```
column_1    0.153884
column_2    0.113498
column_3    0.119478
column_4    0.723147
column_5    0.234896
column_6    0.877742
Name: 1, dtype: float64
```

As we can see from the output, there is a difference between the output of new_df.loc[2] and new_df.iloc[2].

new_df.loc[2] selects the row with label 2 from the DataFrame new_df. Since the DataFrame was reindexed, the label 2 corresponds to the row that was originally labeled 3 in df1. Thus, the output displays the values for that row.

On the other hand, new_df.iloc[2] selects the row with index position 2 from the DataFrame new_df. Since the DataFrame was reindexed, the row at index position 2 corresponds to the row that was originally at index position 1 in df1. Thus, the output displays the values for that row.

4 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'

[13]:
```
df1.mean()
```

[13]:
```
column_1    0.628004
column_2    0.525699
column_3    0.301697
column_4    0.659606
column_5    0.295262
column_6    0.557184
dtype: float64
```

[14]:
```
df1['column_2'].std()
```

[14]:
```
0.3113971489102172
```

5 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.]

[15]:
```
df1.loc['column_2', 1]= "sonal"
```

[16]:
```
df1['column_2'].mean()
```

[16]:
```
0.5256989416371213
```

To compute mean() of the data, all the data should be same type and numeric value data. Since we have added "string" data type in between float data type compiler gets confused and unable to compute mean.

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

In pandas, a window function (also known as a rolling or sliding window) is a function that operates on a specified window of data in a time series or DataFrame. The window moves over the data in a specified direction, such as forward or backward, and calculates a statistic or value for each window.

The different types of window functions available in pandas are:

Rolling: This function creates a rolling window object that can be used to perform calculations on a rolling window of data. The window can be specified using the window size, which can be either a number of periods or a time frequency.

Expanding: This function creates an expanding window object that can be used to perform calculations on an expanding window of data. The window starts from the first period and includes all preceding periods.

Exponentially weighted: This function creates an exponentially weighted moving window object that can be used to perform calculations on a moving window of data. The window size can be specified using the span or the center of mass, which controls the rate at which the weights decrease over time.

Rolling with time-based offsets: This function creates a rolling window object that uses a time-based offset to define the window size. The window can be specified using a time frequency, such as hours, days, or weeks.

Rolling with variable-length windows: This function creates a rolling window object that has a variable-length window size. The window size can be defined based on the values in the data or using a function that calculates the window size.

These window functions can be used to perform various calculations on time series data, such as calculating rolling means, cumulative sums, and exponential moving averages. They are powerful tools for analyzing and visualizing time series data in pandas.

7 Q7. Write a code to print only the current month and year at the time of answering this question.

[Hint: Use pandas.datetime function]

[1]:
```
import pandas as pd

# get the current date and time
current_date = pd.datetime.now()

# extract the month and year from the current date
current_month = current_date.month
current_year = current_date.year

# print the month and year
print("Current month: ", current_month)
print("Current year: ", current_year)
```

```
Current month:  7
Current year:  2023
```

```
/tmp/ipykernel_856/1498775486.py:4: FutureWarning: The pandas.datetime class is
deprecated and will be removed from pandas in a future version. Import from
datetime module instead.
  current_date = pd.datetime.now()
```

8 Q8. Write a Python program that takes in two dates as input (in the format YYYY-MM-DD) and calculates the difference between them in days, hours, and minutes using Pandas time delta. The program should prompt the user to enter the dates and display the result.

[7]:
```
import pandas as pd

# prompt the user to enter the dates
date1 = input("Enter the first date (YYYY-MM-DD): ")
date2 = input("Enter the second date (YYYY-MM-DD): ")

# convert the dates to datetime objects
date1 = pd.to_datetime(date1)
date2 = pd.to_datetime(date2)

# calculate the difference between the dates
diff = date2 - date1

# extract the number of days, hours, and minutes from the difference
days = diff.days
hours = diff.seconds // 3600
minutes = (diff.seconds % 3600) // 60

# display the result
print(f"The difference between {date1.date()} and {date2.date()} is {days}␣
 ↪days, {hours} hours,and {minutes} minutes")
```

```
Enter the first date (YYYY-MM-DD): 2003/08/13
Enter the second date (YYYY-MM-DD): 1975/11/25
The difference between 2003-08-13 and 1975-11-25 is -10123 days, 0 hours,and 0
minutes
```

9 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.

[ ]:
```
import pandas as pd

# prompt the user to enter the file path, column name, and category order
file_path = input("Enter the file path: ")
col_name = input("Enter the column name: ")
```

```
categories = input("Enter the category order (comma-separated): ").split(",")

# read the CSV file
df = pd.read_csv(file_path)

# convert the specified column to categorical data type
df[col_name] = pd.Categorical(df[col_name], categories=categories, ordered=True)

# display the sorted data
print(df.sort_values(by=[col_name]))
```

```
Enter the file path:
Enter the column name:
```

10 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.

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

# prompt the user to enter the file path
file_path = input("Enter the file path: ")

# read the CSV file
df = pd.read_csv(file_path)

# create a pivot table to group the data by product category and year
pivot = pd.pivot_table(df, values='month_number', index='total_units',␣
 ↪columns='total_profit', aggfunc='sum')

# plot the stacked bar chart
pivot.plot(kind='bar', stacked=True)

# set the chart title and axis labels
plt.title('Sales by Product Category')
plt.xlabel('Year')
plt.ylabel('Sales')

# display the chart
plt.show()
```

11 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 following:

```
* Prompt the user to enter the file path of the CSV file containing the student data
*  Read the CSV file into a Pandas DataFrame
*   Calculate the mean, median, and mode of the test scores using Pandas tools
*  Display the mean, median, and mode in a table.



## Assume the CSV file contains the following columnsM
*    Student ID: The ID of the studentR
 * 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
```

[ ]:
```python
import pandas as pd

# prompt the user to enter the file path
file_path = input("Enter the file path of the CSV file containing the student␣
 ↪data: ")

# read the CSV file into a Pandas DataFrame
df = pd.read_csv(file_path)
```

```
# calculate the mean, median, and mode of the test scores
mean = df['gre'].mean()
median = df['gre'].median()
mode = df['gre'].mode().tolist()

# display results in table format
print('+-----------+--------+')
print('| Statistic | Value  |')
print('+-----------+--------+')
print(f'| Mean      | {mean:.1f}  |')
print(f'| Median    | {median}  |')
print(f'| Mode      | {mode}  |')
print('+-----------+--------+')
```

[ ]:

[ ]: