Assignment

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})

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

Ans:can use the `iloc` indexer to access rows by their integer location. In Python, indexing is zero-based, so the second row has an index of 1. Here's the code to print the data present in the second row of the DataFrame `df`:

python

Copy code

```
import        as

              'Data Science'   'Machine Learning'   'Big Data'   'Data Engineer'
           2  3  6  4
                    'course_name'                'duration'


                         1
print
```

This will output:

vbnet

Copy code

```
         3
Name: 1         object
```

In this output, you can see the data present in the second row, where "Machine Learning" is the course_name, and 3 is the duration.

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

Ans:In Pandas, both `loc` and `iloc` are used for indexing and selecting data from a DataFrame, but they have some key differences:

`iloc` (Integer-location based indexing):

- `iloc` is primarily integer-based indexing, meaning you use integer indices to locate elements.
- It is useful when your DataFrame has integer indices, and you want to select data based on the row and column positions.
- You can use integer indices or slices for both rows and columns.
- The syntax is `df.iloc[row_index, column_index]`.

Example:

python

Copy code

```
1  0
```

`loc` (Label-based indexing):

- `loc` is label-based indexing, which means you use labels (index or column names) to locate elements.
- It is useful when your DataFrame has non-integer indices or you want to select data based on labels rather than positions.
- You can use labels or slices for both rows and columns.
- The syntax is `df.loc[row_label, column_label]`.

Example:

python

Copy code

```
1  'course_name'
```

In summary:

- `iloc` is used for integer-location based indexing.
- `loc` is used for label-based indexing.

It's important to choose the appropriate method based on your specific use case and the type of indices (integer or labels) your DataFrame has.

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].

Ans:It appears there is an issue with the quotation marks in your code. Python uses straight quotes (`'` or `"`) for string literals. Here's the corrected code:
python

Copy code

```
import        as

               'Data Science'   'Machine Learning'   'Big Data'   'Data Engineer'
            2   3   6   4
                        'course_name'                'duration'


           3   0   1   2



print             2


print              2
```

Now, this should work as expected. The corrected code reindexes the DataFrame using the specified order and then prints the result for both `loc` and `iloc` based on the reindexed DataFrame.

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)
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’
Ans:Here's the code to find the specified statistical measurements for the DataFrame `df1`:
python

Copy code

```
import        as
import       as

           'column_1'  'column_2'  'column_3'  'column_4'  'column_5'  'column_6'
           1  2  3  4  5  6

                                6  6



print "Mean of each column:"
print

                       'column_2'
print "\nStandard deviation of column 'column_2':"
```

This code uses Pandas and NumPy to create a random DataFrame (`df1`) and then calculates the mean of each column as well as the standard deviation of the 'column_2'.

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.]
Ans:Here's the code to replace the data in the second row of 'column_2' with a string variable and then find the mean of 'column_2':
python

Copy code

```
import        as
import       as

           'column_1'  'column_2'  'column_3'  'column_4'  'column_5'  'column_6'
           1  2  3  4  5  6

                                6  6

        2  'column_2'    'string_data'
```

```
                        'column_2'
print "Mean of 'column_2' after replacement:"
```

This code first replaces the data in the second row of 'column_2' with the string `'string_data'` using `df1.loc[2, 'column_2']`, and then calculates the mean of 'column_2' using `df1['column_2'].mean()`.

If you get an error during execution, it might be due to the fact that replacing numerical data with a string may lead to inconsistencies in data types, and operations like mean may not work as expected. In such cases, it's generally advisable to maintain consistent data types within a column.

Q6. What do you understand about the windows function in pandas and list the types of windows
functions?
Ans:In Pandas, the term "window functions" refers to operations that are applied on a set of data points within a specified window. These functions are particularly useful for time-series data and other ordered data where you want to perform calculations over a moving or expanding window.

Types of Window Functions in Pandas:

Rolling Windows:

- The rolling window functions operate on a specified window of consecutive data points.
- Common rolling window functions include:
  - `rolling.mean()`: Calculates the rolling mean.
  - `rolling.sum()`: Calculates the rolling sum.
  - `rolling.min()`, `rolling.max()`: Calculate the rolling minimum and maximum.
  - `rolling.std()`: Calculates the rolling standard deviation.
  - `rolling.var()`: Calculates the rolling variance.
- Example:
- python
- Copy code

```
'rolling_mean'       'column_1'                3
```

-

Expanding Windows:

- The expanding window functions consider all data points from the start of the series up to the current point.
- Common expanding window functions include:
  - `expanding.mean()`: Calculates the expanding mean.
  - `expanding.sum()`: Calculates the expanding sum.
  - `expanding.min()`, `expanding.max()`: Calculate the expanding minimum and maximum.
  - `expanding.std()`: Calculates the expanding standard deviation.
  - `expanding.var()`: Calculates the expanding variance.
- Example:
- python
- Copy code

```
'expanding_mean'       'column_1'
```

-

Exponential Moving Windows (EWMA):

- EWMA is a type of window function that assigns exponentially decreasing weights to the data points.
- It is often used to give more weight to recent observations.
- Common EWMA functions include:
  - `ewm.mean()`: Calculates the exponentially weighted moving average.
  - `ewm.min()`, `ewm.max()`: Calculate the exponentially weighted moving minimum and maximum.
  - `ewm.std()`: Calculates the exponentially weighted moving standard deviation.
  - `ewm.var()`: Calculates the exponentially weighted moving variance.
- Example:
- python
- Copy code

```
'ewma_mean'       'column_1'           3
```

-

These window functions provide flexibility in analyzing and aggregating data over different time periods, allowing for trend analysis, volatility estimation, and more.

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

[Hint: Use pandas.datetime function]
Ans:However, it's important to note that the `pandas.datetime` function has been deprecated, and it's recommended to use `pandas.to_datetime` for more recent versions of Pandas. Here's the code using `pandas.to_datetime`:

python

Copy code

```
import        as
import          as


                                  'now'


                                                    '%Y-%m'
print "Current month and year:"
```

This code uses `pd.to_datetime('now')` to get the current date and time, and then `strftime('%Y-%m')` is used to format it to display only the current month and year.

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.

ans:Here's a Python program that takes in two dates as input, in the format 'YYYY-MM-DD', calculates the difference between them in days, hours, and minutes using Pandas time delta, and displays the result:

python

Copy code

```
import        as


def calculate_time_difference
 try
```

```
                divmod                          3600
            divmod          60

print "\nTime Difference:"
print f"Days: {days}"
print f"Hours: {hours}"
print f"Minutes: {minutes}"

except
print "Invalid date format. Please enter dates in the format YYYY-MM-DD."


          input "Enter the first date (YYYY-MM-DD): "
          input "Enter the second date (YYYY-MM-DD): "
```

This program uses `pd.to_datetime` to convert the input date strings to Pandas datetime objects, then calculates the time difference using Pandas time delta. The result is displayed in terms of days, hours, and minutes. If the input date format is invalid, the program provides an appropriate message.

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.

ans:Below is a Python program that reads a CSV file containing categorical data, converts a specified column to a categorical data type, and displays the sorted data. The program prompts the user to enter the file path, column name, and category order:

python

Copy code

```
import        as


def process_categorical_column
 try
```

```
        True


print "\nSorted Data:"
print

except
print "File not found. Please check the file path."
except
print "Column not found. Please check the column name."


            input "Enter the file path (CSV): "
              input "Enter the column name to convert to categorical: "
                 input "Enter the category order (comma-separated): "         ','
```

This program uses `pd.read_csv` to read the CSV file, converts the specified column to a categorical data type using `pd.Categorical`, and then displays the sorted data. The user is prompted to enter the file path, column name, and category order. Note that the category order should be entered as a comma-separated list.

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.

Ans:Below is a Python program that reads a CSV file containing sales data for different products, visualizes the data using a stacked bar chart to show the sales of each product category over time. The program prompts the user to enter the file path and displays the chart:

python

Copy code

```
import        as
import                    as
```

```
def visualize_sales_data
 try



 if 'time' in            and 'product_category' in            and 'sales' in


                          'time'          'product_category'         'sales'


                  'bar'          True
           'Sales of Each Product Category Over Time'
            'Time'
            'Sales'


 else
 print "Required columns not found in the DataFrame. Please check the CSV file
structure."

 except
 print "File not found. Please check the file path."


            input "Enter the file path (CSV): "
```

This program uses `pd.read_csv` to read the CSV file and then checks if the DataFrame contains the necessary columns (`time`, `product_category`, and `sales`). If the columns are present, it pivots the DataFrame and creates a stacked bar chart using `matplotlib`. The user is prompted to enter the file path, and the chart is displayed.

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
Ans:Here's a Python program that meets the specified requirements:
python

Copy code

```
import        as


def analyze_student_data
 try



 if 'Student ID' in            and 'Test Score' in
```

```
                'Test Score'
                  'Test Score'
                'Test Score'



 'Statistic'   'Mean'  'Median'  'Mode'
 'Value'                          ', '      map str



 print "\nSummary Table:"
 print

 else
 print "Required columns not found in the DataFrame. Please check the CSV file
structure."

 except
 print "File not found. Please check the file path."


            input "Enter the file path of the CSV file containing the student
data: "
```

This program follows the specified steps:

1. Prompt the user to enter the file path of the CSV file.
2. Read the CSV file into a Pandas DataFrame.
3. Calculate the mean, median, and mode of the 'Test Score' column.
4. Display the results in a table format.

Make sure that the CSV file structure matches the expected columns ('Student ID' and 'Test Score').