1.Explain three-dimensional data indexing?

Ans: Definition Semantic queries on indexed objects allow the reuse of the 3D scenes.Definition Semantic queries on indexed objects allow the reuse of the 3D scenes.Introduction Nowadays, the 3D is a highly expanding media. More particularly with the emergence of dedicated standards such as VRML and X3D, 3D animations are widely used on the Web. The continuous evolution of computing capabilities of desktop computers is also a factor that facilitates the large deployment of 3D information contents. At the same time the demand in term of 3D information is becoming more and more sustained in various domains such as spatial planning, risks management, telecommunications, transports, defense, and tourism. 3D information should represent a real world scene as accurately as possible and should exhibit properties (like, topological relations) to allow complex spatial analysis [1]. The construction of a 3D scene is a complex and time consuming task. Thus, being able to reuse the 3D scenes is a very important issue for the multimedia.Introduction Nowadays, the 3D is a highly expanding media. More particularly with the emergence of dedicated standards such as VRML and X3D, 3D animations are widely used on the Web. The continuous evolution of computing capabilities of desktop computers is also a factor that facilitates the large deployment of 3D information contents. At the same time the demand in term of 3D information is becoming more and more sustained in various domains such as spatial planning, risks management, telecommunications, transports, defense, and tourism. 3D information should represent a real world scene as accurately as possible and should exhibit properties (like, topological relations) to allow complex spatial analysis [1].The construction of a 3D scene is a complex and time consuming task. Thus, being able to reuse the 3D scenes is a very important issue for the multimedia.

2. what is the difference between a series and a dataframe.?

Ans:. While a data stream is received over time, so it is a time series in that sense, it does not necessarily encode data that will be analyzed as a time series. The time a specific piece of data is received may not represent the time of measurement, moreover, time may not be an important feature of the data.Usually when people speak of time series data, they have in mind past observations that are already stored, not real time processing of a data stream. The defining feature is that the order of the data is the focus of the analysis (in fact, it need not even be ordered by time, the same techniques work for other orderings).

So a data stream can have time series data or non time series data. If it has time series data, it will usually be analyzed by real time techniques that differ from standard time series techniques for stored data.Series is a type of list in pandas which can take integer values, string values, double values and more. … Series can only contain single list with index, whereas dataframe can be made of more than one series or we can say that a dataframe is a collection of series that can be used to analyse the data

3. what role does pandas play in data cleaning.?

Ans:. Data Cleaning Using Python Pandas

A comprehensive guide to using built-in Pandas functions to clean data prior to analysis.

Introduction:

Over time companies produce and collect a massive amount of data, depending on the company this can come in many different forms such as user-generated content, job applicant data, blog posts, sensor data and payroll transactions. Due to the immense number of source systems that can generate data and the number of people that contribute to data generation we can never guarantee that the data we are receiving is a clean record. These records may be incomplete due to missing attributes, they may have an incorrect spelling for user-entered text fields or they may have an incorrect value such as a date of birth in the future.

As a data scientist, it's important that these data quality issues are recognised early during our exploration phase and cleansed prior to any analysis. By allowing uncleaned data through our analysis tools we run the risk of incorrectly representing companies or users data by delivering poor quality findings based on incorrect data. Today we will be using Python and Pandas to explore a number of built-in functions that can be used to clean a dataset.

Getting Started

For today's article, we are using PyCharm which is an integrated development environment built for Python. For beginners its an excellent tool to use as it streamlines the creation of virtual environments and the installation of specific Python packages such as Pandas. Using virtual environments allows you to manage your project dependencies without impacting your operating systems default Python installation.

Begin by opening Pycharm and selecting File > New Project from the navigation bar. Here you can name your project using Location and either create a new virtual environment or reference an existing one.

PyCharm's New Project dialogue box showing where to name the project and where the virtual environment are created.

PyCharm's 'New Project' dialogue box showing where to name a project and virtual environment creation paths.

To get you started we have created a base Python script below and a CSV file both of which will be referenced throughout the article. Both of these files can be download and saved to your root project directory. The final script can be found here.

Python snippet that reads a CSV file, casts specific columns to default data types and renames a column.

The above script demonstrates a number of DataFrame manipulations after reading a file into memory. On lines 5–7, we are overriding a number of Pandas default configurations which when dealing with larger files can clip the console output when printing. To read the data into memory we use Pandas built-in function read_csv() on line 10 which takes a file name as a parameter. On lines 13–15, we set the data type of three columns which has a number of benefits. Firstly, setting the data type improves performance when processing DataFrame rows by reducing the memory footprint. Secondly, it enriches

the descriptive statistics output we get when running Pandas .describe() built-in function. On line 18, we perform a column rename which is commonly practised to convert a generic column name to something more meaningful.

Console output after calling Pandas built-in .isnull().sum() function.

If we need to know the percentage of the null elements as a percentage of the whole dataset then we can do the following.

Python snippet to calculate the percentage of missing elements as a whole of the dataset.

Removing Columns

One element that jumps out after calling .info() and .isnull().sum() is the tax_file_no which across 1,000 records has 1,000 null values. The easiest way to remove these types of rows is by using Pandas .dropna(). The .dropna() function takes the form .dropna(axis=0, how='any', thresh=None, subset=None, inplace=False). The axis parameter determines whether the function is applied to rows axis='index' or columns axis='columns'. The how parameter can be how='any' or how='all', which means that the column or row can be dropped if any or all elements have null values.

Removing Rows

After checking for completely null columns it's worth checking to see if there are any rows that do not contain enough usable elements. We can achieve this by making use of .dropna(thresh=2) to remove any rows that have less than two elements.

Python snippet showing how to check rows that will be removed that have less than two elements.

On line 2, we drop any rows that have less than two elements. Line 3 then returns any rows from the original DataFrame whose index does not exist in under_threshold_removed i.e. dropped rows. It accomplishes this by negating ~ pandas built-in .index.isin() function. In our example, one record is returned that only contains the employee number.

Filling Missing Values

In certain circumstances, we may want to retain rows that contain missing values and instead give them a default value when missing. For this example, we are going to map the null values within the gender column to U for Unknown. Pandas provide a built-in function that can achieve this .fillna(value=None, method=None, axis=None, inplace=False, limit=None, downcast=None).

4. How do you pandas to make a data frame out of n-dimensional arrays.?

Ans:. Pandas Dataframe

The simple datastructure pandas.DataFrame is described in this article. It includes the related information about the creation, index, addition and deletion. The text is very detailed.

In short: it's a two-dimensional data structure (like table) with rows and columns.

Related course: Data Analysis with Python Pandas

Create DataFrame

What is a Pandas DataFrame

Pandas is a data manipulation module. DataFrame let you store tabular data in Python.

The DataFrame lets you easily store and manipulate tabular data like rows and columns.

A dataframe can be created from a list (see below), or a dictionary or numpy array (see bottom).

Create DataFrame from list

You can turn a single list into a pandas dataframe:

```
Import pandas as pd

Data = [1,2,3]

Df = pd.DataFrame(data)
```

The contents of the dataframe is then:

```
>>> df

  0

    1   1
    2   1  2
    3   3
    4   >>>
```

Before the contents, you'll see every element has an index (0,1,2).

This works for tables (n-dimensional arrays) too:

```
Import pandas as pd

Data = [['Axel',32], ['Alice', 26], ['Alex', 45]]

Df = pd.DataFrame(data,columns=['Name','Age'])
```

This outputs:

```
>>> df

   Name  Age

0   Axel   32

1  Alice   26

2   Alex   45

>>>
```

5.Explain the notion of pandas plotting.?

Ans:. Plotting

Pandas uses the plot() method to create diagrams.

Example

Import pyplot from Matplotlib and visualize our DataFrame:

```
Import pandas as pd

Import matplotlib.pyplot as plt

Df = pd.read_csv('data.csv')

Df.plot()

Plt.show()
```

Scatter Plot

Specify that you want a scatter plot with the kind argument:

Kind = 'scatter'

A scatter plot needs an x- and a y-axis.

In the example below we will use "Duration" for the x-axis and "Calories" for the y-axis.

Include the x and y arguments like this:

X = 'Duration', y = 'Calories'

Example

```
Import pandas as pd

Import matplotlib.pyplot as plt

Df = pd.read_csv('data.csv')

Df.plot(kind = 'scatter', x = 'Duration', y = 'Calories')

Plt.show()
```