1. Attribute types:

Attribute types refer to the different kinds of characteristics or properties that can be associated with data. In data science, attribute types are important for understanding and analyzing data. Common attribute types include numerical (e.g., age, height), categorical (e.g., gender, color), ordinal (e.g., rating scale), and binary (e.g., yes/no). Knowing the attribute types helps determine appropriate analysis techniques and data preprocessing steps.

2. Similarity and Dissimilarity:

Similarity and dissimilarity are measures used to compare the likeness or differences between objects or data points. In data science, similarity and dissimilarity metrics play a crucial role in tasks like clustering, recommendation systems, and data mining. Similarity measures quantify how similar two objects are, while dissimilarity measures quantify how different or dissimilar they are.

3. Data Preprocessing:

Data preprocessing involves transforming and preparing raw data before it can be used for analysis or modeling. It includes tasks like data cleaning (removing errors or inconsistencies), data integration (combining data from multiple sources), data transformation (scaling or normalizing data), and handling missing values. Data preprocessing is essential for ensuring data quality and improving the effectiveness of subsequent analysis.

4. Traits of Big Data:

Big data refers to large and complex datasets that cannot be easily managed with traditional data processing techniques. The traits of big data are often referred to as the "3Vs": Volume (large amount of data), Velocity (data generated at high speed), and Variety (data in various formats and types). Big data also poses challenges in terms of storage, processing, analysis, and privacy. Data scientists need specialized tools and techniques to handle big data effectively.

5. Reporting vs Analysis:

Reporting and analysis are two different aspects of data science. Reporting involves summarizing and presenting data in a structured format, such as dashboards or reports, to provide insights and answer specific questions. Analysis, on the other hand, involves exploring and investigating data to discover patterns, relationships, and trends. It often involves using statistical methods, machine learning algorithms, and visualization techniques to gain deeper insights and make data-driven decisions.

6. NumPy:

NumPy is a powerful Python library used for numerical computing. It provides efficient and high-performance tools for working with arrays and matrices. NumPy's main data structure is the ndarray (n-dimensional array), which allows for fast mathematical operations on large datasets. NumPy also offers a wide range of mathematical functions and methods for array manipulation, broadcasting, and linear algebra operations.

7. List vs Arrays:

In Python, a list is a built-in data structure that can hold elements of different data types and can grow or shrink dynamically. It is versatile but may not offer optimal performance for numerical computations. On the other hand, an array, specifically a NumPy array, is a fixed-size and homogeneous data structure designed for efficient mathematical operations. Arrays provide faster execution and take up less memory compared to lists when working with numerical data.

8. Numpy Functions:

NumPy provides a wide range of functions for various mathematical operations. These functions include basic mathematical calculations (e.g., addition, subtraction, multiplication) as well as more advanced operations like trigonometric functions, exponential functions, statistical functions, and linear algebra operations. NumPy functions are optimized for speed and efficiency, making them ideal for handling large numerical datasets.

9. Binary Functions:

Binary functions, in the context of NumPy, are functions that operate on two input arrays and produce a single output array. These functions perform element-wise operations on corresponding elements of the input arrays. Examples of binary functions in NumPy include addition (np.add), subtraction (np.subtract), multiplication (np.multiply), division (np.divide), and logical operations like AND, OR, XOR.

10. Boolean Functions:

Boolean functions in NumPy are functions that operate on arrays and return Boolean (True/False) values based on certain conditions. These functions allow for element-wise comparisons and logical operations on arrays. Examples of boolean functions in NumPy include greater than (np.greater), less than (np.less), equal to (np.equal), logical AND (np.logical_and), logical OR (np.logical_or), and logical NOT (np.logical_not). Boolean functions are commonly used for filtering, indexing, and conditional operations in data science tasks.

11. Input-Output File Organization:

Input-output (I/O) file organization refers to how data is structured and stored in files when reading from or writing to external storage devices. In data science, efficient file organization is crucial for handling large datasets and ensuring smooth data operations. Some commonly used file organization techniques include CSV (Comma-Separated Values) for tabular data, JSON (JavaScript Object Notation) for semi-structured data, and database management systems (e.g., MySQL, PostgreSQL) for structured data. Parquet and HDF5 are file formats optimized for large datasets, offering efficient compression and storage.

12. Pandas: Basics of Pandas

Pandas is a powerful Python library widely used in data science for data manipulation, analysis, and cleaning. It provides data structures and functions that make working with structured and tabular data more efficient. The two

main data structures in Pandas are the Series and DataFrame objects. A Series is a one-dimensional labeled array, while a DataFrame is a two-dimensional table-like structure. Indexes in Pandas are labels or identifiers associated with rows or columns, allowing for easy data retrieval and manipulation.

13. Series Object:

A Series object in Pandas is a one-dimensional labeled array that can hold data of any type. It is similar to a column in a spreadsheet or a database table. Each element in a Series is associated with an index label, which can be customized or assigned automatically. Series objects provide various methods and operations for data analysis, manipulation, and computation. They can be created from lists, arrays, or dictionaries, and are often used to represent and work with a single variable or data column.

14. DataFrame Object:

A DataFrame object in Pandas is a two-dimensional labeled data structure resembling a table with rows and columns. It is the primary data structure used for data manipulation and analysis in Pandas. DataFrames are highly flexible and efficient, allowing for easy handling of structured data. They can be created from various data sources like CSV files, Excel spreadsheets, or other Pandas data structures. DataFrames provide methods for data cleaning, filtering, aggregation, merging, and statistical analysis. They are widely used for exploratory data analysis and preparing data for modeling or visualization.

15. Indexes:

Indexes in Pandas are labels or identifiers associated with rows or columns in a DataFrame or Series. Indexes provide a way to uniquely identify and access data within a Pandas object. By default, Pandas assigns a numeric index starting from zero to each row in a DataFrame or element in a Series. However, you can also customize the index to use labels, such as dates, names, or any other meaningful identifier. Indexing operations allow for efficient data selection and manipulation based on the specified index labels. Indexes play a crucial role in data alignment, merging/joining of datasets, and efficient data retrieval in Pandas.