# 21CSE221T - Big Data Tools and Techniques

## CT3 Key

## Part A

**Q1) Types of NoSQL ( 1 mark each)**

Four main types of NoSQL databases:

- Document databases - stores data in JSON, BSON, or XML documents
- Key-value stores - Every data element in the database is stored as a key value pair consisting of an attribute name (or "key") and a value.
- Column-oriented databases - a column store is organized as a set of columns. This means that when you want to run analytics on a small number of columns, you can read those columns directly without consuming memory with the unwanted data.
- Graph databases - A graph database focuses on the relationship between data elements. Each element is stored as a node (such as a person in a social media graph). The connections between elements are called links or relationships.

**Q2) Stream vs batch processing (1 mark each point)**

| S.No | Batch Processing | Stream Processing |
|---|---|---|
| 1 | Processing based on the data collected over time | Processing based on immediate data for instant result is called Real-time Processing. |
| 2 | dealing with “data at rest,” | Deals with “data in motion” |
| 3 | Store and then process pattern | Process and store if required pattern |
| 4 | Can be slow and can take much time | Low latency and always must be available |

**Q3) Any four zookeeper commands out of the below: (1 mark each command)**

### Syntax

```
create /path /data
```

### Sample

```
create /FirstZnode “Myfirstzookeeper-app”
```

### Output

```
[zk: localhost:2181(CONNECTED) 0] create /FirstZnode “Myfirstzookeeper-app”
Created /FirstZnode
```

To create a **Sequential znode**, add **-s flag** as shown below.

### Syntax

```
create -s /path /data
```

### Sample

```
create -s /FirstZnode second-data
```

### Output

```
[zk: localhost:2181(CONNECTED) 2] create -s /FirstZnode “second-data”
Created /FirstZnode0000000023
```

To create an **Ephemeral Znode**, add **-e flag** as shown below.

### Syntax

```
create -e /path /data
```

### Sample

```
create -e /SecondZnode “Ephemeral-data”
```

### Output

```
[zk: localhost:2181(CONNECTED) 2] create -e /SecondZnode “Ephemeral-data”
Created /SecondZnode
```

Remember when a client connection is lost, the ephemeral znode will be deleted. You can try it by quitting the ZooKeeper CLI and then re-opening the CLI.

## Get Data

It returns the associated data of the znode and metadata of the specified znode. You will get information such as when the data was last modified, where it was modified, and information about the data. This CLI is also used to assign watches to show notification about the data.

### Syntax

```
get /path
```

### Sample

```
get /FirstZnode
```

### Output

```
[zk: localhost:2181(CONNECTED) 1] get /FirstZnode
“Myfirstzookeeper-app”
cZxid = 0x7f
ctime = Tue Sep 29 16:15:47 IST 2015
mZxid = 0x7f
mtime = Tue Sep 29 16:15:47 IST 2015
pZxid = 0x7f
cversion = 0
dataVersion = 0
aclVersion = 0
ephemeralOwner = 0x0
dataLength = 22
```

```
numChildren = 0
```

To access a sequential znode, you must enter the full path of the znode.

## Sample

```
get /FirstZnode0000000023
```

## Output

```
[zk: localhost:2181(CONNECTED) 1] get /FirstZnode0000000023
“Second-data”
cZxid = 0x80
ctime = Tue Sep 29 16:25:47 IST 2015
mZxid = 0x80
mtime = Tue Sep 29 16:25:47 IST 2015
pZxid = 0x80
cversion = 0
dataVersion = 0
aclVersion = 0
ephemeralOwner = 0x0
dataLength = 13
numChildren = 0
```

**Q4) Three traditional DWH systems and key features of DWH systems ( 2+2 marks)**

Examples: Teradata, HP Vertica, Exalytics

Features:

- Fully functional
- Query optimization
- Handles enterprise workload
- Supports advanced features like in memory computing and filtering

Q5)

**Discrete vs Continuous data 2 marks**

- Discrete variables can only take on certain values within a range.
- Between zero and ten, there are eleven discrete values for variables.
- There will be no 2.3 or 5.78 because we don’t count these numbers when counting to 10.
- Continuous numbers can take on any value e.g. real numbers.
- In opposition to the counting numbers, there are an infinite number of real numbers between zero and ten.

**Bar plot vs histogram 2 marks**

Bar plot – Categorical data in x axis and numerical data in y axis

Histogram – The distribution of one continuous variable, the bins cannot be interchanged

## Part A

**Q6) Spark and Data Structures (Spark - 4 mark , DS – 6 marks, 3 each)**

Apache Spark is a unified analytics engine for large-scale data processing. It provides high level APIs like Spark SQL, Spark Streaming, MLib, and GraphX to allow interaction with core functionalities of Apache Spark. Spark also facilitates several core data abstractions on top of the distributed collection of data which are RDDs, DataFrames, and DataSets.

## Spark RDD (Resilient Distributed Dataset)

Spark RDD stands for **Resilient Distributed Dataset** which is the core data abstraction API and is available since very first release of Spark (**Spark 1.0**). It is a lower-level API for manipulating distributed collection of data. To enable fault tolerance, RDD uses **DAG (Directed Acyclic Graph)** which consists of a set of vertices and edges. The vertices and edges in DAG represent the RDD and the operation to be applied on that RDD respectively. The transformations defined on RDD are lazy and executes only when an action is called. Let's have a quick look at features and limitations of RDD:

*RDD Features:*

1. **Immutable collection:** RDD is an immutable partitioned collection distributed on different nodes. A partition is a basic unit of parallelism in Spark. The immutability helps to achieve fault tolerance and consistency.
2. **Distributed data:** RDD is a collection of distributed data which helps in big data processing by distributing the workload to different nodes in the cluster.
3. **Lazy evaluation:** The defined transformations do not gets evaluated until an action is called. It helps Spark in optimizing the overall transformations in one go.
4. **Fault tolerant:** RDD can be recomputed in case of any failure using DAG(Directed acyclic graph) of transformations defined for that RDD.
5. **Multi-language support:** RDD APIs supports **Python, R, Scala, and Java** programming languages.

*Limitation of RDD:*

1. **No optimization engine:** RDD does not have an in-built optimization engine. Programmers need to write their own code in order to minimize the memory usage and to improve execution performance.

### Spark DataFrame

Spark 1.3 introduced two new data abstraction APIs – **DataFrame and DataSet**. The DataFrame APIs organizes the data into named columns like a table in relational database. It enables programmers to define schema on a distributed collection of data. Each row in a DataFrame is of object type row. Like an SQL table, each column must have same number of rows in a DataFrame. In short, DataFrame is lazily evaluated plan which specifies the operations needs to be performed on the distributed collection of the data. DataFrame is also an immutable collection. Below are the features and limitations of DataFrame:

*DataFrame Features:*

1. **In-built Optimization:** DataFrame uses **Catalyst** engine which has an in-built execution optimization that improves the data processing performance significantly. When an action is called on a DataFrame, the Catalyst engine analyzes the code and resolves the references. Then, it creates a logical plan. After that, the created logical plan gets translated into an optimized physical plan. Finally, this physical plan gets executed on the cluster.
2. **Hive compatible:** The DataFrame is fully compatible with Hive query language. We can access all hive data, queries, UDFs, etc using Spark SQL from hive MetaStore and can execute queries against these hive databases.
3. **Structured, semi-structured, and highly structured data support:** DataFrame APIs supports manipulation of all kind of data from structured data files to semi-structured data files and highly structured parquet files.
4. **Multi-language support:** DataFrame APIs are available in **Python, R, Scala, and Java**.
5. **Schema support:** We can define a schema manually or we can read a schema from a data source which defines the column names and their data types.

*DataFrame Limitations:*

1. **Type safety:** Each row in a DataFrame is of object type row and hence is not strictly typed. That is why DataFrame does not support compile time safety.

## Spark DataSet

As an extension to the DataFrame APIs, **Spark 1.3** also introduced DataSet APIs which provides strictly typed and object-oriented programming interface in Spark. It is immutable, type-safe collection of distributed data. Like DataFrame, DataSet APIs also uses Catalyst engine in order to enable execution optimization. DataSet is an extension to the DataFrame APIs. Features and limitations of the DataSet are as below:

*DataSet Features:*

1. **Combination of RDD and DataFrame:** DataSet enables functional programming like RDD APIs and relational queries and execution optimization like DataFrame APIs. Thus, it provides the benefit of best of both worlds – RDDs and DataFrames.

2. **Type-safe:** Unlike DataFrames, DataSet APIs provides compile time type safety. It conforms the specification at compile time using defined case classes (for Scala) or Java beans (for Java).

*DataSet Limitations:*

1. **Limited language support:** DataSet is only available to JVM based langauges like Java and Scala. Python and R do not support DataSet because these are dynamically typed languages.
2. **High garbage collection:** JVM types can cause high garbage collection and object instantiation cost.

**Q7) MongoDB Commands**

i) and ii) **5 marks**

db.createCollection("bookreviews")

Book_ID, Book_Name, Review_Score, Bucket, Publisher, Publisher_Avg_Score, Author_Avg_Score

db.bookreviews.insert([

{

Book_ID: 123,

Book_Name: ‘Mastering Apache Storm’,

Review_Score: 4,

Bucket: B,

Publisher: “Packet Publishing”,

Publisher_Avg_Score: 4,

Author_Avg_Score:3

},

{

Book_ID: 456,

Book_Name: ‘Java Recipes’,

Review_Score: 4,

Bucket: A,

Publisher: “O’Reilly”,

Publisher_Avg_Score: 5,

Author_Avg_Score:3

}

])

iii) db.bookreviews.find({Publisher_Avg_Score >= 4}).pretty() **2 marks**

iv) db. bookreviews.update({'Publisher':' Packet Publishing '}, {$set:{ 'Publisher':' Pckt Publishing ''}},{multi:true}) **3 marks**

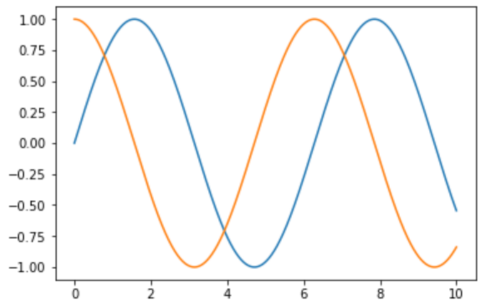
**Q8) challenges associated with Big Data Analytics and the roadmap to enterprise analytical success**

- **general challenges associated with Big Data analytics are as follows: 3 marks**
    - Nearly every company is investing in Big Data, machine learning, and AI
    - Often, the company has a corporate mandate
    - Finding the right use cases can be challenging
    - Even after you *find them, the outcome may be uncertain*
    - Even after you *achieve them, whether or not the optimal targets have been* identified can be elusive (for example, when using HDFS for storing only data)
- **Roadmap to enterprise success - General guidelines for data science and analytics initiatives7 marks**
    - **Conduct meetings and one-on-one reviews with business partners in the** organization to review their workflows and get feedback on where analytics and/or data mining would provide the most value
    - **Identify specific aspects of business operations that are important and related** to the firm's *revenue stream; the use case would have a measurable impact once* completed
    - The use cases do not have to be *complex; they can be simple tasks, such as ML or* Data Mining
    - Intuitive, easily understood, you can explain it to friends and family
    - Ideally the use case takes effort to accomplish today using conventional means. The solution should not only benefit a **range of users, but also have executive visibility**
    - Identify **Low Difficulty - High Value (Short) vs High Difficulty - High Value** (**Long) use cases**
    - Educate business sponsors, share ideas, show **enthusiasm (like a long job** interview)
    - Score **early wins for Low Difficulty - High Value, create Minimum Viable Solutions**, and get management to buy in before further enhancing the use solutions developed. (takes time)

**Q9) With code snippets, explain how to visualize the data using the below charts in Python**

i) Line chart **2 marks**

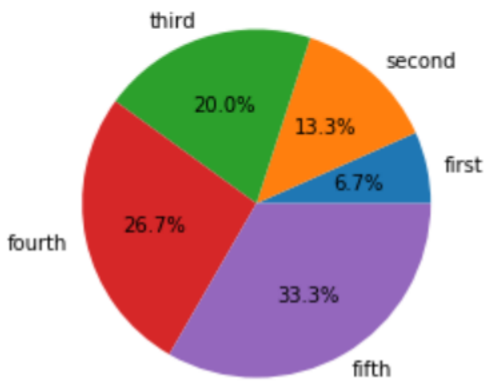
```python
import numpy as np
import matplotlib.pyplot as plt

x = np.linspace(0, 10, 1000)

plt.plot(x,np.sin(x))
plt.plot(x, np.cos(x))
plt.show()
```



- Used to represent relation between two data on two different axes X and Y
- allow looking at the behavior of one or several variables over time and identifying the trends.
- Presents information as series of data points
- Suitable for continuous, can represent discrete data also
- Stacked line charts

ii) Pie chart **2 marks**

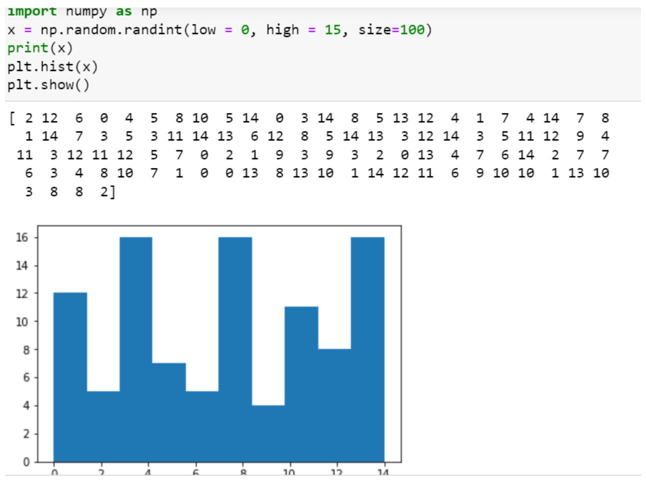
```python
import matplotlib.pyplot as plt
import numpy as np


x = [100, 200, 300, 400, 500]
labels = ['first', 'second', 'third', 'fourth', 'fifth']

plt.pie(x,labels=labels,autopct='%1.1f%%')

plt.show()
```



- Pie charts show the components of the whole.
- Companies that work with both traditional and big data may use this technique to look at customer segments or market shares.

- The difference lies in the sources from which these companies take raw data for the analysis.

iii) Histogram **3 marks**

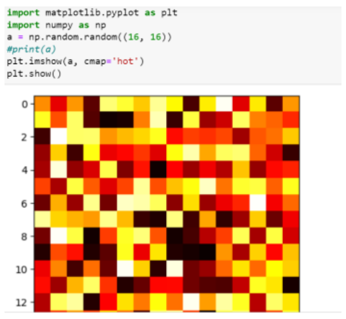


```
import numpy as np
x = np.random.randint(low = 0, high = 15, size=100)
print(x)
plt.hist(x)
plt.show()
```

```
[ 2 12  6  0  4  5  8 10  5 14  0  3 14  8  5 13 12  4  1  7  4 14  7  8
  1 14  7  3  5  3 11 14 13  6 12  8  5 14 13  3 12 14  3  5 11 12  9  4
 11  3 12 11 12  5  7  0  2  1  9  3  9  3  2  0 13  4  7  6 14  2  7  7
  6  3  4  8 10  7  1  0  0 13  8 13 10  1 14 12 11  6  9 10 10  1 13 10
  3  8  8  2]
```

- A histogram is a graphical representation of data points organized into user-specified ranges.
- Similar in appearance to a bar graph, the histogram condenses a data series into an easily interpreted visual by taking many data points and grouping them into logical ranges or bins.

iv) Heatmap **3 marks**



```
import matplotlib.pyplot as plt
import numpy as np
a = np.random.random((16, 16))
#print(a)
plt.imshow(a, cmap='hot')
plt.show()
```

- A heat map is a two-dimensional representation of data in which values are represented by colors.
- The variation in color may be by hue or intensity
- Giving obvious visual cues to the reader about how the phenomenon is clustered or varies over space.