SREENIVASA INSTITUTE OF TECHNOLOGY AND MANAGEMENT STUDIES
(AUTONOMOUS)
MCA DEPARTMENT

LECTURE NOTES

Subject Name: BIG DATA ANALYTICS

Year / Branch: II MCA – I SEMESTER

Regulation: R24

Prepared By: Mr. N P Gangadhar,

Assistant Professor

II MCA – I SEMESTER			
COURSE CODE:	**24MCA213**	**CREDITS:**	**4**
COURSE TITLE:	**BIG DATA ANALYTICS**	**L-T-P:**	**4-0-0**

PREREQUISITES: *Courses on "Database Management Systems" , "Object Oriented Programming through JAVA" and knowledge on Intelligence Techniques.*

COURSE EDUCATIONAL OBJECTIVES*:*

CEO1 *To explore the fundamental concepts of Big Data.*
CEO2 *To Learn Basic concepts of Hadoop.*
CEO3 *To Write Hadoop MapReduce Programs for analyzing Big data.*
CEO4 *To Explore Hadoop Ecosystem.*

UNIT - I: UNDERSTANDING BIG DATA — **Lecture Hrs:10**

What is Big Data, Concepts and Terminology - Datasets, Data Analysis, Data Analytics, Big Data Characteristics – volume, velocity, variety, veracity, value, Different Types of Data – Structured Data, Unstructured Data, Semi-Structured Data, Case Study Background.

UNIT - II : HADOOP BASICS — **Lecture Hrs:10**

Brief history of Hadoop, Apache hadoop and the hadoop ecosystem. A weather dataset, analyzing the data with unix tools, analyzing the data with hadoop , Understanding different Hadoop modes, understanding Hadoop Features-Understanding HDFS, Understanding MapReduce, Learning the HDFS and Mapreduce Architecture-Understanding the HDFS architecture, Understanding the MapReduce Architecture,
Understanding the HDFS and MapReduce architecture by plot.

UNIT - III: WORKING WITH PIG AND HIVE — **Lecture Hrs:12**

Pig -Execution Types, An Example, Pig Latin-Structure, Statements,Types, Schemas, Functions, Data Processing Operators.Hive – An example, Tables –Managed Tables and External tables, Partitions and Buckets, Importing data, Altering Data, Dropping Tables, Querying Data-Sorting and Aggragating,
Mapreduce Scripts.

UNIT - IV : INTRODUCTION TO APACHE SPARK — **Lecture Hrs:12**

What is Apache Spark, Characteristics – Speed, Easy of Use, Modularity, Extensibility, Apache Spark Components as a Unified Stack, Spark Basic Data Types, Spark Core and SQL- Dynamic Partition Pruning, SQL Operations, Schemas and Creating Data Frames

UNIT - V: HBASE, ZOOKEEPER, SQOOP — **Lecture Hrs:12**

HBase Overview – Limitations of Hadoop, what is HBase,HBase and HDFS,StorageMechansim in HBase,Features of HBase, Applications of HBase. ZooKeeper Overview – what is ZooKeeper, Distributed Application, Benefits of Distributed Applications,Challenges of Distributed Applications, What is Apache
Zookeeper meant for, Benefits of ZooKeeper

TEXT BOOKS:

1. *Big Data Fundamentals: Concepts, Drivers & Techniques", 1/e, 2016, Thomas Erl, Wajid Khattak, Paul Buhler, Prentice Hall.*
2. *"Hadoop:The Definitive Guide," 3/e, 2012, Tom White, O'REILLY Publications.*
3. *"Learning Spark"*

REFERENCE BOOKS:

1. *Taming the Big Data Tidal Wave: Finding Opportunities in Huge Data Streams with Advanced Analytics , 2012, Bill Franks, John Wiley & Sons..*

2. *Understanding Big Data: Analytics for Enterprise Class Hadoop and Streaming Data, 2012, Paul C. Zikopoulos, Chris Eaton, Dirk deRoos, Thomas Deutsch, George Lapis, McGraw-Hill.*

3. *Intelligent Data Analysis, 2007, Michael Berthhold, David J.Hand, Springer.*

4. *Understanding Big Data: Analytics for Enterprise Class Hadoop and streaming data, 2011, Paul Zikopoulos, Chris Eaton, Paul Zikopoulos, McGraw hill.*

5. *Big Data for Dummies, 2012, Hurwitz, Alan Nugent, Dr. Fern Halper, Marcia Kaufman,John Wiley & Sons*

COURSE OUTCOMES: *On successful completion of this course, students will be able to:*		POs related to COs
CO1	**Realize** characteristics of Big Data and various types of data like structured, unstructured and semistructured	PO1,PO2
CO2	**Understand** two major components of Hadoop	PO1,PO2,PO3,PO4, PO5
CO3	**Analyze** Big data using Hadoop Map Reduce programs	PO1, PO2,PO4, PO5
CO4	get **Acquainted** with two Data Access Components of Hadoop Ecosystem called pig and hive	PO1,PO3, PO4
CO5	Acquire **Knowledge** on Data Storage Component of Hadoop Ecosystem called Hbase, Data Integration Component of Hadoop Ecosystem like sqoop and Monitoring, Management and Orchestration of Hadoop Ecosystem component like zookeeper	PO1,PO2, PO3,PO4

CO-PO MAPPING(DETAILED; HIGH:3; MEDIUM:2; LOW:1)

Course	**POs** / **COs**	**PO1**	**PO2**	**PO3**	**PO4**	**PO5**	**PO6**	**PO7**	**PO8**
C303: Big Data Analytics	**C303.1**	**3**	**2**	-	-	-	-	-	-
	C303.2	**3**	**3**	**2**	**3**	**2**	-	-	-
	C303.3	**3**	**3**	-	**3**	**3**	-	-	-
	C303.4	**3**	-	**3**	**3**	-	-	-	-
	C303.5	**3**	**3**	**2**	**3**	-	-	-	-
	C303	**3**	**2.75**	**2.3**	**3**	**2.5**	-	-	-

UNIT IV - INTRODUCTION TO APACHE SPARK

Unit Overview

This unit introduces Apache Spark, a powerful open-source distributed computing framework designed for fast Big Data processing. Students will learn the features and characteristics of Spark, including speed, ease of use, modularity, and extensibility. The unit explains Spark's unified architecture, core components, basic data types, Spark Core, Spark SQL, Dynamic Partition Pruning (DPP), SQL operations, schemas, and DataFrame creation. By the end of this unit, students will understand how Apache Spark enables high-performance data analytics and machine learning applications.

Learning Outcomes

After completing this unit, students will be able to:

1. Explain the concept and architecture of Apache Spark.
2. Describe the characteristics of Spark.
3. Identify the components of the Spark Unified Stack.
4. Explain Spark's basic data types.
5. Understand Spark Core and Spark SQL.
6. Explain Dynamic Partition Pruning (DPP).
7. Perform SQL operations using Spark SQL.
8. Create schemas and DataFrames in Spark.
9. Apply Spark concepts for Big Data processing and analytics.

Importance of Studying this Unit

Apache Spark is one of the most widely used Big Data processing frameworks because it performs computations much faster than Hadoop MapReduce. It supports real-time analytics, machine learning, graph processing, and streaming applications. Spark is extensively used by organizations such as Netflix, Uber, Amazon, Facebook, and Microsoft for large-scale data analysis.

Key Concepts

Apache Spark, Spark Core, Spark SQL, Spark Streaming, MLlib, GraphX, DataFrame, Dataset, RDD, Dynamic Partition Pruning (DPP), Schema, Distributed Computing, In-Memory Computing, Lazy Evaluation, Cluster Computing.

What is Apache Spark

- Apache Spark is an open-source, distributed computing framework designed for big data processing and analytics.
- Designed to process large-scale datasets efficiently across multiple machines.
- Provides both batch and real-time (streaming) data processing.
- Runs programs up to 100x faster in memory and 10x faster on disk than traditional Hadoop MapReduce.
- It offers high-level APIs in **Scala, Java, Python (PySpark), R, and SQL**.

Apache Spark Characteristics

Apache Spark stands out as a fast, flexible, and unified data processing framework. Its key characteristics are:

- ✓ Speed
- ✓ Ease of Use
- ✓ Modularity
- ✓ Extensibility
- ✓ Fault Tolerance
- ✓ Scalability

Speed

- In-memory computation makes Spark up to 100x faster than Hadoop MapReduce (which writes intermediate results to disk).
- Uses Directed Acyclic Graph (DAG) execution engine to optimize task scheduling.
- Efficient for iterative algorithms (like ML training) where intermediate results can be reused.

Ease of Use

- APIs available in Scala, Java, Python (PySpark), and R.
- Developers can use SQL queries (Spark SQL) or high-level APIs like DataFrames and Datasets.
- Provides interactive shells (spark-shell, pyspark) for quick exploration.

Modularity

Comes with specialized libraries that integrate seamlessly:

- Spark SQL → for structured data & queries.
- Spark Streaming / Structured Streaming → for real-time data.

- MLlib → for machine learning.
- GraphX → for graph computation.

All libraries run on the **same Spark engine**, making it a unified stack.

Extensibility

- Supports various storage systems: HDFS, Hive, Cassandra, MongoDB, Amazon S3, Azure Data Lake, etc.
- Reads/writes multiple formats: Parquet, ORC, JSON, Avro, CSV.
- Integrates with cluster managers like YARN, Kubernetes, and Mesos.
- Works with data lake formats like Delta Lake, Apache Hudi, and Iceberg.

Fault Tolerance

- Uses RDD (Resilient Distributed Dataset) to recover lost partitions automatically.
- Lineage information allows recomputation of lost data.
- Structured Streaming provides exactly-once guarantees

Scalability

- Scales from a single laptop to thousands of cluster nodes.
- Handles terabytes to petabytes of data.
- Dynamic resource allocation allows flexible cluster usage.

Apache Spark Components as a Unified Stack

Apache Spark provides a unified framework for handling different types of big data workloads (batch, streaming, ML, and graphs) under one system. This avoids the need for multiple tools.

Here are the main components:

Spark Core (Foundation)

- The base engine of Apache Spark.
- Provides essential functionalities like:
    - Memory management
    - Task scheduling
    - Fault tolerance (RDDs & lineage)
    - Interacting with storage systems (HDFS, S3, Cassandra, HBase, etc.)
- Defines the **RDD (Resilient Distributed Dataset)** abstraction.
    - Immutable distributed collection of data.
    - Provides transformations (map, filter, flatMap) and actions (count, collect).
    - Fault-tolerance via lineage tracking (recomputes lost partitions).
- All other Spark libraries are built on top of Spark Core.

Spark SQL (Structured Data Processing)

- Its Provides DataFrames (table-like structure with named columns).
- Provides Datasets (type-safe, available in Scala/Java).

- Includes Catalyst Optimizer for intelligent query optimization.
- Uses Tungsten Execution Engine for efficient memory usage and code generation.
- Supports SQL queries, HiveQL, and integration with BI tools.
- Reads/writes multiple formats: JSON, Parquet, ORC, Avro, JDBC.

Ex:

```
df = spark.read.json("data.json")

df.createOrReplaceTempView("people")

result = spark.sql("SELECT name, age FROM people WHERE age > 30")

result.show()
```

MLlib (Machine Learning Library)

- Spark's scalable machine learning library for big data.
- Provides distributed ML algorithms:
    - Classification (Logistic Regression, Decision Trees).
    - Regression (Linear Regression).
    - Clustering (KMeans, Gaussian Mixture).
    - Collaborative Filtering (ALS for recommendations).
    - Dimensionality Reduction (PCA, SVD).
- Includes utilities for **feature extraction, transformation, selection, and pipeline building**.
- Integrates with Spark SQL and DataFrames.

Example: Building a Logistic Regression model.

```
from pyspark.ml.classification import LogisticRegression


training = spark.read.format("libsvm").load("data/mllib/sample_libsvm_data.txt")

lr = LogisticRegression(maxIter=10, regParam=0.01)

model = lr.fit(training)
```

GraphX (Graph Processing)

- A library for graph computation and analysis.
- Built on top of RDDs.
- Provides an API for graph-parallel computation.
- Allows transformation of graphs (e.g., subgraph extraction, join operations).
- Includes graph algorithms:
    - PageRank (used by Google Search).
    - Connected Components.
    - Triangle Counting.

- Shortest Paths.

Why It's Called a Unified Stack

- All components share the same Spark Core engine → no need to switch tools for different workloads.
- Example end-to-end pipeline:
  - Use Structured Streaming to capture live user activity.
  - Store data in Parquet/Delta Lake.
  - Query with Spark SQL.
  - Train ML models with MLlib.
  - Run GraphX for network analysis.
- This makes Spark an all-in-one platform for batch, streaming, machine learning, and graph analytics.

Apache Spark Architecture

Apache Spark follows a master–slave architecture with a driver program (master) and cluster workers (slaves).
It uses a distributed execution engine to process large datasets across multiple nodes in a cluster.

Components of Spark Architecture

Driver Program (Master Node)

- The **entry point** of a Spark application.
- Runs the **main() function** of your program.
- Responsible for:
  - Maintaining **SparkContext** (connection to cluster).
  - Converting user code into a **Directed Acyclic Graph (DAG)**.
  - Splitting the DAG into **stages** and **tasks**.
  - Scheduling tasks on executors.
  - Collecting results back from executors.

Cluster Manager

- Manages resources (CPU, memory) across the cluster.
- Spark supports multiple cluster managers:
  - **Standalone** (built-in Spark manager)
  - **YARN** (Hadoop cluster manager)
  - **Apache Mesos**

- **Kubernetes**

Worker Nodes

- The machines in the cluster that **run tasks assigned by the driver**.
- Each worker node hosts:

a) Executor

- A process launched for each Spark application.
- Runs tasks and stores data in memory/disk.
- Executors remain alive throughout the application (unlike MapReduce).

b) Task

- The **smallest unit of execution** in Spark.
- One stage is divided into many tasks, distributed across executors.

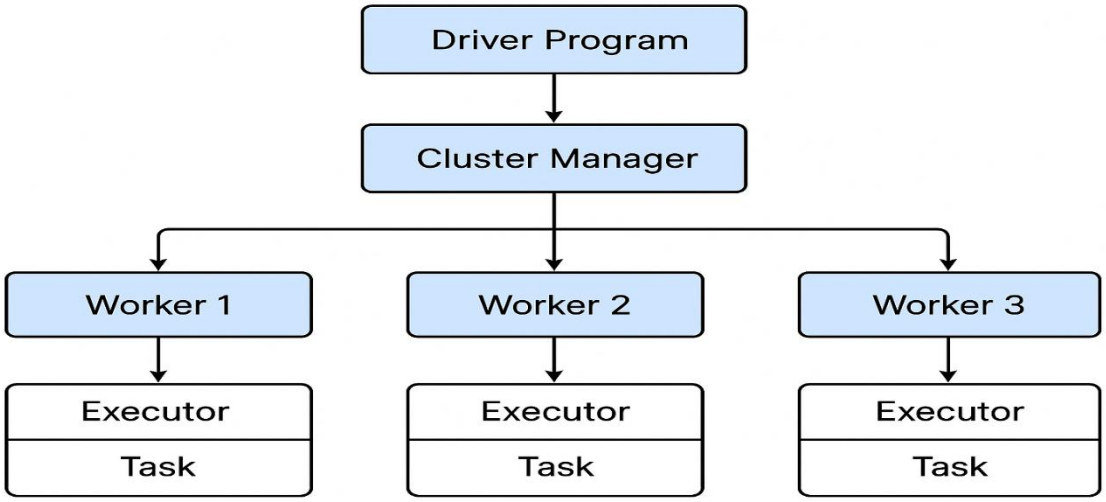
RDD (Resilient Distributed Dataset)

- The fundamental abstraction in Spark.
- Immutable distributed collection of objects, partitioned across nodes.
- Provides **fault tolerance** using **lineage information** (recomputes lost partitions).

How Spark Executes a Job (Workflow)

1. **User Program**: You write Spark code (Scala, Python, Java, R, SQL).
2. **Driver Program**:
    - Initializes SparkContext.
    - Converts user code into a **DAG (Directed Acyclic Graph)** of stages.
3. **DAG Scheduler**:
    - Divides DAG into **stages**.
    - Each stage is split into **tasks** (based on data partitions).
4. **Task Scheduler**:
    - Assigns tasks to available executors on worker nodes.
5. **Cluster Manager**:
    - Provides resources (CPU, memory).
6. **Executors (on Workers)**:
    - Execute the tasks in parallel.
    - Store intermediate results in memory/disk.

7. **Driver**: Collects and combines the results.



Spark Basic Data Types

Spark SQL and DataFrames support the following data types

Numeric types

- **Byte Type:** Represents 1-byte signed integer numbers. The range of numbers is from -128 to 127.
- **Short Type:** Represents 2-byte signed integer numbers. The range of numbers is from -32768 to 32767.
- **Integer Type:** Represents 4-byte signed integer numbers. The range of numbers is from -2147483648 to 2147483647.
- **Long Type:** Represents 8-byte signed integer numbers. The range of numbers is from -9223372036854775808 to 9223372036854775807.
- **Float Type:** Represents 4-byte single-precision floating point numbers.
- **Double Type:** Represents 8-byte double-precision floating point numbers.
- **Decimal Type:** Represents arbitrary-precision signed decimal numbers.

String type

- **String Type:** Represents character string values.
- **Varchar Type(length):** A variant of StringType which has a length limitation. Data writing will fail if the input string exceeds the length limitation. Note: this type can only be used in table schema, not functions/operators.

- **Char Type(length):** A variant of VarcharType(length) which is fixed length. Reading column of type CharType(n) always returns string values of length n. Char type column comparison will pad the short one to the longer length.

Binary Type: Represents byte sequence values.

Boolean Type: Represents boolean values.

Datetime type

Date Type: Represents values comprising values of fields year, month and day, without a time-zone.

Timestamp Type: Timestamp with local time zone(TIMESTAMP_LTZ). It represents values comprising values of fields year, month, day, hour, minute, and second, with the session local time-zone. The timestamp value represents an absolute point in time.

Example:

```
from pyspark.sql.types import *
# Define a schema using Spark data types
schema = StructType([
    StructField("id", IntegerType(), True),
    StructField("name", StringType(), True),
    StructField("age", IntegerType(), True),
    StructField("salary", DoubleType(), True),
    StructField("skills", ArrayType(StringType()), True),
    StructField("properties", MapType(StringType(), StringType()), True),
    StructField("join_date", DateType(), True)
])
```

Spark Core and SQL

Spark Core:

- All the functionalities being provided by Apache Spark are built on the highest of the Spark Core.
- It delivers speed by providing in-memory computation capability. Spark Core is the foundation of parallel and distributed processing of giant dataset.
- It is the main backbone of the essential I/O functionalities and significant in programming and observing the role of the spark cluster.
- It holds all the components related to scheduling, distributing and monitoring jobs on a cluster, Task dispatching, Fault recovery.
- The functionalities of this component are:
    - It contains the basic functionality of spark. (Task scheduling, memory management, fault recovery, interacting with storage systems).
    - Home to API that defines RDDs.

Spark SQL Structured data:

- The Spark SQL component is built above the spark core and used to provide the structured processing on the data.
- It provides standard access to a range of data sources. It includes Hive, JSON, and JDBC.
- It supports querying data either via SQL or via the hive language. This also works to access structured and semi-structured information.
- It also provides powerful, interactive, analytical application across both streaming and historical data.
- Spark SQL could be a new module in the spark that integrates the relative process with the spark with programming API.
- The main functionality of this module is:
    - It is a Spark package for working with structured data.
    - It Supports many sources of data including hive tablets, parquet, json.
    - It allows the developers to intermix SQK with programmatic data manipulation supported by RDDs in python, scala and java.

DataFrame

A DataFrame is a distributed collection of data organized into rows and named columns, similar to a table in a relational database or an Excel spreadsheet. It is one of the most widely used data structures in Apache Spark because it provides better performance and supports SQL-like operations.

A DataFrame is built on top of RDDs and uses Spark SQL's optimization engine (Catalyst Optimizer) for efficient execution.

Features of DataFrames

- Distributed collection of data
- Organized into rows and columns
- Schema-based structure
- Supports SQL queries
- Optimized using Catalyst Optimizer
- Fault-tolerant
- Supports multiple data sources

Advantages

- Faster than RDDs
- Easy to use
- Supports SQL
- Better memory management
- Automatic query optimization

Creating a DataFrame

```
From pyspark.sql Import SparkSession

spark=SparkSession.builder.appName("DataFrameExample").getOrCreate()

data=[
   (101,"Ram",50000),
   (102,"Hari",60000)
]
```

df=spark.createDataFrame(data,["ID","Name","Salary"])

df.show()

Output

```
+---+----+------+
|ID|Name|Salary|
+---+----+------+
|101|Ram|50000|
|102|Hari|60000|
+---+----+------+
```

Case Study: Real-Time Ride Booking Analysis Using Apache Spark

Background

Uber receives millions of ride requests every day from users across different cities. The system generates massive amounts of real-time data, including ride requests, driver locations, GPS coordinates, payment details, customer ratings, and traffic information. Traditional batch processing systems could not analyze this data quickly enough to support real-time decision-making.

Problem Statement

- Process millions of ride requests in real time.
- Match passengers with nearby drivers instantly.
- Analyze traffic conditions dynamically.
- Detect fraudulent transactions.
- Generate live business analytics.

Unit Highlights

- Apache Spark is an in-memory distributed computing framework for Big Data processing.
- Spark is faster than Hadoop MapReduce due to in-memory computation.

- The four key characteristics of Spark are Speed, Ease of Use, Modularity, and Extensibility.
- The Spark Unified Stack includes Spark Core, Spark SQL, Spark Streaming, MLlib, and GraphX.
- Spark supports multiple programming languages such as Python, Scala, Java, and R.
- Dynamic Partition Pruning (DPP) improves query performance by scanning only relevant partitions.
- DataFrames and Spark SQL simplify structured data processing.
- Apache Spark is widely used for real-time analytics, machine learning, streaming, fraud detection, recommendation systems, and business intelligence.

TEXT BOOKS:

1. Big Data Fundamentals: Concepts, Drivers & Techniques", 1/e, 2016, Thomas Erl, Wajid Khattak, Paul Buhler, Prentice Hall.
2. "Hadoop:The Definitive Guide," 3/e, 2012, Tom White, O'REILLY Publications.
3. "Learning Spark"

REFERENCE BOOKS:

1. Taming the Big Data Tidal Wave: Finding Opportunities in Huge Data Streams with Advanced Analytics, 2012, Bill Franks, John Wiley & Sons..
2. Understanding Big Data: Analytics for Enterprise Class Hadoop and Streaming Data, 2012, Paul C. Zikopoulos, Chris Eaton, Dirk deRoos, Thomas Deutsch, George Lapis, McGraw-Hill.
3. Intelligent Data Analysis, 2007, Michael Berthhold, David J.Hand, Springer.
4. Understanding Big Data: Analytics for Enterprise Class Hadoop and streaming data, 2011, Paul Zikopoulos, Chris Eaton, Paul Zikopoulos, McGraw hill.
5. Big Data for Dummies, 2012, Hurwitz, Alan Nugent, Dr. Fern Halper, Marcia Kaufman,John Wiley & Sons

S.No	Questions
UNIT IV- INTRODUCTION TO APACHE SPARK	
PART-A (Two Marks Questions)	
1.	What is Apache Spark?
2.	What are the main advantages of Apache Spark over Hadoop MapReduce?
3.	Define Resilient Distributed Dataset (RDD).
4.	What are DataFrames and Datasets in Spark?
5.	Why is Spark considered easy to use?
6.	What is modularity in Apache Spark?
7.	How does Apache Spark achieve high speed in data processing?
8.	Name the major components of Spark's unified stack.
9.	What is the function of Spark Core?
10.	What is Spark SQL used for?
11.	What is the purpose of Spark Streaming?
12.	List any four basic data types supported in Spark.
13.	What is Dynamic Partition Pruning (DPP) in Spark SQL?
14.	Why is Dynamic Partition Pruning important?
15.	What is a schema in Spark?
16.	How can you create a DataFrame with an explicit schema?
PART-B (Ten Marks Questions)	
1.	Explain Apache Spark. How is it different from Hadoop MapReduce?
2.	Describe the architecture of Apache Spark with its key components.
3.	What are RDDs, DataFrames, and Datasets? Compare them with examples.
4.	Explain the different components of Apache Spark as a unified stack.
5.	Discuss the role of Spark Core in managing distributed data processing.
6.	Describe the basic data types supported in Spark SQL.
7.	Explain the concept of Dynamic Partition Pruning in Spark SQL.
8.	Compare Static Partition Pruning and Dynamic Partition Pruning in Spark.
9.	Explain the process of running SQL queries on Spark DataFrames.
10.	Discuss how Spark integrates SQL queries with structured and semi-structured data sources.
11.	What is a schema in Spark? Why is schema enforcement important?
12.	Explain different ways to create DataFrames in Spark (RDDs, external files, schema definition).