SRM UNIVERSITY DELHI-NCR, SONEPAT, HARYANA

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Bachelor of Technology(B.Tech)

Big Data Analytics, Tools And Techniques Level III (21CS0302)

NAME	: Ritish Bhatia
REGISTER NO	: 10322210050
SEMESTER	: 6th Semester
YEAR	: 3rd Year

SRM University Delhi-NCR, Sonepat, Haryana, Rajiv Gandhi Education City, Delhi-NCR, Sonepat-131029, Haryana (India)

SRM UNIVERSITY DELHI-NCR, SONEPAT, HARYANA

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that is a bonafide work done by Mr. Ritish Bhatia for the **(21CS0302) Big Data Analytics, Tools and Techniques Level III** as a part of the **B.Tech-CSE (Core),** course in **SRM University Delhi-NCR, Sonepat, Haryana** during the year of **2024-25**. The record was found to be completed and satisfactory.

HOD/Coordinator **Subject In-Charge**

Submitted for the Practical Examination held on ___________________

Internal Examiner **External Examiner**

INDEX

Experiment – 01

Aim : What is big data? What are types of big data?

Introduction:

Big data is a large and complex collection of data that is difficult to process using traditional data management tools, characterized by high volume, velocity, and variety.

Key Characteristics (6Vs of Big Data)

1. **Volume** – The vast amount of data generated.
2. **Velocity** – The speed at which data is created and processed.
3. **Variety** – Different types of data (structured, unstructured, semi-structured).
4. **Veracity** – The reliability and accuracy of data.
5. **Value** – The usefulness and insights derived from data.
6. **Variability** – The inconsistencies and fluctuations in data.

Types of Big Data:

1. Structured Data

- Organized in a fixed format, easy to store and retrieve.
- Stored in relational databases (SQL) with predefined schemas.
- **Examples:**
  - Customer transaction records in an e-commerce platform.
  - Employee payroll stored in an Excel sheet.
  - Banking data in SQL databases.

2. Unstructured Data

- Lacks a predefined format, making storage and processing more challenging.
- Requires advanced tools like AI, NLP, and image processing.
- **Examples:**
  - Social media posts, emails, and customer reviews.
  - Videos, audio recordings, and satellite images.

- Web pages and scanned documents.

3. Semi-structured Data

- Contains elements of both structured and unstructured data.
- Does not follow a rigid schema but has metadata to provide some organization.
- **Examples:**
    - JSON and XML files used in web applications.
    - NoSQL databases (MongoDB, Cassandra) storing flexible datasets.
    - Sensor logs from IoT devices.

Big Data is essential in fields like finance, healthcare, marketing, and artificial intelligence, helping businesses make data-driven decisions.

Experiment – 02

Aim: Extracting textual data from an unsecured and a secure website

Introduction:

In the era of big data, extracting textual data from websites is crucial for data analysis, market research, sentiment analysis, and various machine learning applications. This process, known as web scraping, involves retrieving and processing data from both unsecured (HTTP) and secure (HTTPS) websites.

Extract Data from an Unsecured Website (HTTP)

- Send a request to an HTTP website.
- Parse the response to extract textual data.
- **Example code:**

```
import requests
from bs4 import BeautifulSoup

url_http = "http://example.com"
# Replace with an actual unsecured website

try:
response = requests.get(url_http)
response.raise_for_status()  # Check for errors
soup = BeautifulSoup(response.text, "html.parser")
print(soup.get_text())
except requests.exceptions.RequestException as e:
print(f"Error fetching data: {e}")
```

Extract Data from a Secure Website (HTTPS)

- Send a request to an HTTPS website and extract text.
- **Example code:**

```
url_https = https://example.com
  # Replace with an actual secure website
```

```
try:
    response = requests.get(url_https, headers={"User-Agent": "Mozilla/5.0"})
    response.raise_for_status()
    soup = BeautifulSoup(response.text, "html.parser")
    print(soup.get_text())
except requests.exceptions.RequestException as e:
    print(f"Error fetching data: {e}")
```

Result: Data is successfully extracted from a secured and unsecured website.

Experiment - 03

Aim: To perform matrix multiplication using Map-Reduce Method in Big Data Analytics**.**

Theory: This experiment demonstrates **matrix multiplication** using the **MapReduce** approach. The **Mapper** distributes matrix elements, and the **Reducer** performs multiplication and sums the results. This method is useful for processing large datasets in **Big Data Analytics**.

Code Implementation:

```
#!/usr/bin/env python3
import sys
from collections import defaultdict

# Simulated input data (Matrix A and B)
input_data = [
    "A,0,0,5", "A,0,1,3", "A,1,0,2",
    "B,0,0,4", "B,1,0,6"
]

# Step 1: Mapper Function
mapped_data = []
for line in input_data:
    matrix, row, col, value = line.split(",")
    row, col, value = int(row), int(col), float(value)

    if matrix == "A":
        for k in range(2):  # Assuming Matrix B has 2 columns
            mapped_data.append((f"{row},{k}", f"A,{col},{value}"))
    else:
        for i in range(2):  # Assuming Matrix A has 2 rows
            mapped_data.append((f"{i},{col}", f"B,{row},{value}"))


# Step 2: Shuffle & Sort (Simulating Hadoop's process)
```

```
grouped_data = defaultdict(list)
for key, value in mapped_data:
    grouped_data[key].append(value)

# Step 3: Reducer Function
result = {}
for key, values in grouped_data.items():
    a_values, b_values = {}, {}

    for val in values:
        mat, index, num = val.split(",")
        index, num = int(index), float(num)

        if mat == "A":
            a_values[index] = num
        else:
            b_values[index] = num

    result[key] = sum(a_values[k] * b_values[k] for k in a_values if k in
b_values)

# Step 4: Output Results
for key, value in result.items():
    print(f"{key}\t{value}")
```

Result : The matrices are successfully multiplied using map-reduce method.

Output:

```
0,0    38.0
1,0    8.0
```

Experiment – 04

Aim: To perform matrix multiplication using PySparks in Big Data Analytics.

Theory:
Matrix multiplication in PySpark is performed using RDD transformations such as map(), join(), and reduceByKey(). The matrices are represented as RDDs with tuples (row_index, col_index, value). The multiplication involves joining elements based on shared indices, computing products, and aggregating results to form the final matrix.

To achieve this:

1. **Reformat Matrix A** so that columns become keys for joining.
2. **Reformat Matrix B** so that rows become keys for joining.
3. **Perform an Inner Join** on the shared index.
4. **Multiply Corresponding Values** and store them with new row-column indices.
5. **Use reduceByKey()** to sum up values for each position in the final matrix.

Code Implementation :

```
from pyspark.sql import SparkSession

# Initialize Spark Session
spark =
SparkSession.builder.appName("MatrixMultiplication").getOrCreate(
)

# Define Matrices as RDDs
matrixA = [
    (0, 0, 2), (0, 1, 3),
    (1, 0, 4), (1, 1, 1)
]
```

```
matrixB = [
    (0, 0, 5), (0, 1, 2),
    (1, 0, 3), (1, 1, 4)
]



rddA = spark.sparkContext.parallelize(matrixA)
rddB = spark.sparkContext.parallelize(matrixB)

# Reformat for Multiplication
rddA = rddA.map(lambda x: (x[1], (x[0], x[2])))  # Key by column
index of A
rddB = rddB.map(lambda x: (x[0], (x[1], x[2])))  # Key by row index
of B

# Perform Multiplication and Sum Results
result = rddA.join(rddB) \
             .map(lambda x: ((x[1][0][0], x[1][1][0]), x[1][0][1] *
x[1][1][1])) \
             .reduceByKey(lambda x, y: x + y)

# Display Results
print("Resultant Matrix Multiplication Output:")
for res in sorted(result.collect()):
    print(res)

# Stop Spark Session
spark.stop()
```

Result : The matrices are successfully multiplied using PySparks.