# Set1:

1. Implement a Word Count program using Hadoop MapReduce:
   1. Write a MapReduce program to count the occurrences of each word in the file text\_data.txt.

***Mapper.py:***

***#!usr/bin/env python3***

***import sys***

***def mapper():***

***for line in sys.stdin:***

***line=line.strip()***

***words=line.split()***

***for word in words:***

***print(‘%s\t%d’%(word,1)***

***if\_\_name\_\_==’\_\_main\_\_’:***

***mapper()***

***Reducer.py:***

***#!usr/bin/env python3***

***import sys***

***def reducer():***

***current\_word = None***

***current\_count = 0***

***for line in sys.stdin:***

***line = line.strip()***

***word, count = line.split('\t', 1)***

***try:***

***count = int(count)***

***exceptValueError:***

***continue***

***if current\_word == word:***

***current\_count += count***

***else:***

***if current\_word:***

***print(f'{current\_word}\t{current\_count}')***

***current\_word = word***

***current\_count = count***

***if current\_word == word:***

***pass***

***if\_\_name\_\_==”\_\_main\_\_”:***

***reducer()***

* 1. Run the MapReduce job and display the output.

***Cat input.txt|python mapper.py***

***Cat input.txt|python mapper.py|sort |python reducer.py***

1. Load a dataset and perform basic transformations using Pig:
   1. Load the dataset employee\_records.csv with the schema {emp\_id, emp\_name, department, salary} into Pig.

***employee\_data = LOAD 'employee\_records.csv' USING PigStorage(',')***

***AS (emp\_id:int, emp\_name:chararray, department:chararray, salary:float);***

* 1. Filter employees with a salary greater than 80,000.

***high\_salary\_employees = FILTER employee\_data BY salary > 80000;***

* 1. Group the data by department.

***grouped\_by\_department = GROUP employee\_data BY department;***

* 1. Compute the average salary for each department.

***department\_avg\_salary = FOREACH grouped\_by\_department GENERATE group AS department, AVG(employee\_data.salary) AS avg\_salary;***

* 1. Store the result in a new file department\_avg\_salary.

***STORE department\_avg\_salary INTO 'department\_avg\_salary' USING PigStorage(',');***

# SET2

1. Load a dataset and perform basic data transformations in Pig:
   1. Load the dataset customer\_orders.csv with the schema {customer\_id, customer\_name, region, order\_amount} into Pig.

***-- Load the dataset with the schema***

***customer\_orders = LOAD 'customer\_orders.csv' USING PigStorage(',') AS (customer\_id:int, customer\_name:chararray, region:chararray, order\_amount:float);***

* 1. Filter records where the order amount is greater than 1000.

***-- Filter records where order\_amount is greater than 1000***

***high\_value\_orders = FILTER customer\_orders BY order\_amount > 1000;***

* 1. Group the data by region.

***-- Group the data by region***

***grouped\_by\_region = GROUP high\_value\_orders BY region;***

* 1. Calculate the total order amount for each region.

***region\_total\_orders = FOREACH grouped\_by\_region GENERATE group AS region, SUM(high\_value\_orders.order\_amount) AS total\_order\_amount;***

* 1. Store the result in a new dataset called region\_total\_orders.

***-- Store the result in a new file***

***STORE region\_total\_orders INTO 'region\_total\_orders' USING PigStorage(',');***

1. Use advanced Pig operations to perform grouping and filtering:
   1. Load the dataset product\_data.csv with schema {product\_id, product\_name, price, category} into Pig.

***-- Load the dataset with the schema***

***product\_data = LOAD 'product\_data.csv'***

***USING PigStorage(',')***

***AS (product\_id:int, product\_name:chararray, price:double, category:chararray);***

* 1. Group the data by category.

***-- Group the data by category***

***grouped\_by\_category = GROUP product\_data BY category;***

* 1. Calculate the total number of products in each category.

***-- Calculate the total number of products in each category***

***category\_product\_count = FOREACH grouped\_by\_category***

***GENERATE group AS category, COUNT(product\_data) AS product\_count;***

* 1. Filter the products with a price greater than 300.

***-- Filter the products where price is greater than 300***

***filtered\_products = FILTER product\_data BY price > 300;***

* 1. Store the result in a new dataset called filtered\_products.

***-- Store the result in a new file***

***STORE filtered\_products INTO 'filtered\_products' USING PigStorage(',');***

# SET3

1. Create a Hive table and query the data using basic SQL operations:
   1. Create a Hive table named employees with schema {emp\_id, name, age, department, salary}.

***CREATE TABLE employees (***

***emp\_id INT,***

***name STRING,***

***age INT,***

***department STRING,***

***salary FLOAT***

***)***

***ROW FORMAT DELIMITED***

***FIELDS TERMINATED BY ','***

***STORED AS TEXTFILE;***

* 1. Load data from the file employees.csv into the employees table.

***LOAD DATA INPATH 'employees.csv' INTO TABLE employees;***

* 1. Write a query to select all employees from the "HR" department.

***SELECT \* FROM employees WHERE department = 'HR';***

* 1. Write a query to find employees with a salary greater than 70,000.

***SELECT \* FROM employees WHERE salary > 70000;***

* 1. Write a query to calculate the average salary by department.

***SELECT department, AVG(salary) AS average\_salary***

***FROM employees***

***GROUP BY department;***

1. Implement a Word Count program using Hadoop MapReduce:
   1. Write a MapReduce program to count the occurrences of each word in the file novels.txt.

***Mapper.py:***

***#!usr/bin/env python3***

***import sys***

***def mapper():***

***for line in sys.stdin:***

***line=line.strip()***

***words=line.split()***

***for word in words:***

***print(‘%s\t%d’%(word,1)***

***if\_\_name\_\_==’\_\_main\_\_’:***

***mapper()***

***Reducer.***

***#!usr/bin/env python3***

***import sys***

***def reducer():***

***current\_word = None***

***current\_count = 0***

***for line in sys.stdin:***

***line = line.strip()***

***word, count = line.split('\t', 1)***

***try:***

***count = int(count)***

***exceptValueError:***

***continue***

***if current\_word == word:***

***current\_count += count***

***else:***

***if current\_word:***

***print(f'{current\_word}\t{current\_count}')***

***current\_word = word***

***current\_count = count***

***if current\_word == word:***

***pass***

***if\_\_name\_\_==”\_\_main\_\_”:***

***reducer()***

* 1. Run the MapReduce job and display the output.

***Cat novel.txt|python mapper.py***

***Cat novel.txt|python mapper.py|sort |python reducer.py***

# SET4

1. Load a dataset and perform basic transformations using Pig:
   1. Load the dataset employee\_details.csv with the schema {emp\_id, emp\_name, department, experience} into Pig.

***-- Load the dataset with the schema***

***employee\_data = LOAD 'employee\_details.csv'***

***USING PigStorage(',')***

***AS (emp\_id:int, emp\_name:chararray, department:chararray, experience:int);***

* 1. Filter employees with more than 10 years of experience.

***-- Filter employees with more than 10 years of experience***

***experienced\_employees = FILTER employee\_data BY experience > 10;***

* 1. Group the data by department.

***-- Group the data by department***

***grouped\_by\_department = GROUP experienced\_employees BY department;***

* 1. Compute the highest experience level for each department.

-- ***Compute the highest experience level for each department***

***max\_experience\_by\_department = FOREACH grouped\_by\_department***

***GENERATE group AS department, MAX(experienced\_employees.experience) AS max\_experience;***

* 1. Store the result in a new file department\_max\_experience.

-- ***Store the result in a new file***

***STORE max\_experience\_by\_department INTO 'department\_max\_experience' USING PigStorage(',');***

1. Create and query tables in Hive using basic SQL operations:
   1. Create a Hive table named inventory with schema {item\_id, item\_name, quantity, price}.

***CREATE TABLE inventory (***

***item\_id INT,***

***item\_name STRING,***

***quantity INT,***

***price FLOAT***

***)***

***ROW FORMAT DELIMITED***

***FIELDS TERMINATED BY ','***

***STORED AS TEXTFILE;***

* 1. Load data from the file inventory\_data.csv into the Hive table.

***LOAD DATA INPATH 'inventory\_data.csv' INTO TABLE inventory;***

* 1. Write a query to find all items with quantity less than 50.

***SELECT \* FROM inventory WHERE quantity < 50;***

* 1. Write a query to calculate the total value of the inventory (quantity \* price).

***SELECT item\_id, item\_name, quantity, price, (quantity \* price) AS total\_value***

***FROM inventory;***

* 1. Write a query to find the most expensive item in the inventory.

***SELECT \* FROM inventory ORDER BY price DESC LIMIT 1;***

# Sub. Code: P23DS3P6 | Sub. Name: Big Data Management and Analytics Lab | Semester: III

1. Use advanced Pig operations to perform grouping and filtering:
   1. Load the dataset sales\_data.csv with schema {sale\_id, product\_name, sale\_amount, region} into Pig.

***-- Load the dataset sales\_data.csv***

***sales\_data = LOAD 'sales\_data.csv'***

***USING PigStorage(',')***

***AS (sale\_id:int, product\_name:chararray, sale\_amount:float, region:chararray);***

* 1. Group the data by region.

***-- Group the data by region***

***grouped\_by\_region = GROUP sales\_data BY region;***

* 1. Calculate the total sales in each region.

***-- Calculate total sales in each region***

***total\_sales\_by\_region = FOREACH grouped\_by\_region***

***GENERATE group AS region, SUM(sales\_data.sale\_amount) AS total\_sales;***

* 1. Filter the sales where the sale amount is greater than 500.

***filtered\_sales = FILTER sales\_data BY sale\_amount > 500;***

* 1. Store the result in a new dataset called region\_sales\_data.

***STORE filtered\_sales INTO 'region\_sales\_data' USING PigStorage(',');***

1. Create a Hive table and query the data using basic SQL operations:
   1. Create a Hive table named students with schema {student\_id, name, age, grade, subject}.

***CREATE TABLE students (***

***student\_id INT,***

***name STRING,***

***age INT,***

***grade FLOAT,***

***subject STRING***

***)***

***ROW FORMAT DELIMITED***

***FIELDS TERMINATED BY ','***

***STORED AS TEXTFILE;***

* 1. Load data from the file students.csv into the students table.

***LOAD DATA LOCAL INPATH 'students.csv' INTO TABLE students;***

* 1. Write a query to select all students with grades greater than 85.

***SELECT \* FROM students WHERE grade > 85;***

* 1. Write a query to calculate the average grade for each subject.

***SELECT subject, AVG(grade) AS average\_grade***

***FROM students***

***GROUP BY subject;***

* 1. Write a query to find the highest grade in the "Mathematics" subject.

***SELECT MAX(grade) AS highest\_grade***

***FROM students***

***WHERE subject = 'Mathematics';***

**Sub. Code: P23DS3P6 | Sub. Name: Big Data Management and Analytics Lab | Semester: III**

1. Implement a Word Count program using Hadoop MapReduce:

a) Write a MapReduce program to count the occurrences of each word in the file

document.txt.

1. Modify the program to exclude common stop words such as "a", "and", "the".

***#!/usr/bin/env python3***

***import sys***

***# Define common stop words***

***stop\_words = {'a', 'and', 'the', 'in', 'of', 'to', 'is', 'with', 'that', 'on'}***

***def mapper():***

***for line in sys.stdin:***

***line = line.strip()***

***words = line.split()***

***for word in words:***

***word = word.lower() # Convert to lowercase for uniformity***

***if word not in stop\_words: # Exclude stop words***

***print('%s\t%d' % (word, 1))***

***if \_\_name\_\_ == '\_\_main\_\_':***

***mapper()***

***REDUCER():***

***#!/usr/bin/env python3***

***import sys***

***def reducer():***

***current\_word = None***

***current\_count = 0***

***for line in sys.stdin:***

***line = line.strip()***

***word, count = line.split('\t', 1)***

***try:***

***count = int(count)***

***except ValueError:***

***continue***

***if current\_word == word:***

***current\_count += count***

***else:***

***if current\_word:***

***print('%s\t%d' % (current\_word, current\_count))***

***current\_word = word***

***current\_count = count***

***if current\_word == word:***

***print('%s\t%d' % (current\_word, current\_count))***

***if \_\_name\_\_ == '\_\_main\_\_':***

***reducer()***

1. Run the MapReduce job and display the output.

***cat document.txt | python3 mapper.py***

***cat document.txt | python3 mapper.py | sort | python3 reducer.py***

1. Load a dataset and perform basic data transformations in Pig:
   1. Load the dataset customer\_data.csv with the schema {customer\_id, customer\_name, country, age} into Pig.

***-- Load the dataset customer\_data.csv***

***customer\_data = LOAD 'customer\_data.csv'***

***USING PigStorage(',')***

***AS (customer\_id:int, customer\_name:chararray, country:chararray, age:int);***

* 1. Filter records where the age is greater than 40.

***-- Filter customers with age greater than 40***

***filtered\_customers = FILTER customer\_data BY age > 40;***

* 1. Group the data by country.

***-- Group the data by country***

***grouped\_by\_country = GROUP filtered\_customers BY country;***

* 1. Calculate the average age of customers in each country.

***-- Calculate the average age of customers in each country***

***average\_age\_by\_country = FOREACH grouped\_by\_country***

***GENERATE group AS country, AVG(filtered\_customers.age) AS avg\_age;***

* 1. Store the result in a new dataset called country\_avg\_age.

***-- Store the result in a new dataset***

***STORE average\_age\_by\_country INTO 'country\_avg\_age' USING PigStorage(',');***

# Sub. Code: P23DS3P6 | Sub. Name: Big Data Management and Analytics Lab | Semester: III

1. Load a dataset and perform basic transformations using Pig:
   1. Load the dataset transactions.csv with schema {transaction\_id, customer\_name, transaction\_amount, transaction\_date} into Pig.

***-- Load the dataset transactions.csv***

***transactions = LOAD 'transactions.csv'***

***USING PigStorage(',')***

***AS (transaction\_id:int, customer\_name:chararray, transaction\_amount:float, transaction\_date:chararray);***

* 1. Filter transactions where the transaction amount is greater than 1000.

***-- Filter transactions with transaction amount greater than 1000***

***high\_value\_transactions = FILTER transactions BY transaction\_amount > 1000;***

* 1. Group the data by transaction\_date.

***-- Group transactions by transaction\_date***

***grouped\_by\_date = GROUP high\_value\_transactions BY transaction\_date;***

* 1. Calculate the total transaction amount for each day.

***-- Calculate the total transaction amount for each day***

***daily\_totals = FOREACH grouped\_by\_date***

***GENERATE group AS transaction\_date, SUM(high\_value\_transactions.transaction\_amount) AS total\_transaction\_amount;***

* 1. Store the result in a new file daily\_transaction\_totals.

***-- Store the result in a file named daily\_transaction\_totals***

***STORE daily\_totals INTO 'daily\_transaction\_totals' USING PigStorage(',');***

1. Create and query tables in Hive using basic SQL operations:
   1. Create a Hive table named courses with schema {course\_id, course\_name, duration, fee}.

***CREATE TABLE courses (***

***course\_id INT,***

***course\_name STRING,***

***duration INT,***

***fee FLOAT***

***)***

***ROW FORMAT DELIMITED***

***FIELDS TERMINATED BY ',';***

* 1. Load data from the file courses.csv into the Hive table.

***LOAD DATA INPATH '/path/to/courses.csv' INTO TABLE courses;***

* 1. Write a query to find all courses with a duration less than 6 months

***SELECT \* FROM courses***

***WHERE duration < 6;***

* 1. Write a query to calculate the total fee for all courses.

***SELECT SUM(fee) AS total\_fee FROM courses;***

* 1. Write a query to find the course with the highest fee.

***SELECT \* FROM courses***

***ORDER BY fee DESC***

***LIMIT 1;***

**Sub. Code: P23DS3P6 | Sub. Name: Big Data Management and Analytics Lab | Semester: III**

1. Implement a Word Count program using Hadoop MapReduce:

a) Write a MapReduce program to count the occurrences of each word in the file

***#!/usr/bin/env python3***

***import sys***

***def mapper():***

***for line in sys.stdin:***

***# Remove leading and trailing spaces***

***line = line.strip()***

***# Split the line into words***

***words = line.split()***

***# Emit each word with a count of 1***

***for word in words:***

***print(f'{word}\t1')***

***if \_\_name\_\_ == "\_\_main\_\_":***

***mapper()***

***REDUCER():***

***#!/usr/bin/env python3***

***import sys***

***def reducer():***

***current\_word = None***

***current\_count = 0***

***for line in sys.stdin:***

***# Remove leading and trailing spaces***

***line = line.strip()***

***# Parse the input we got from the mapper***

***word, count = line.split('\t', 1)***

***try:***

***count = int(count)***

***except ValueError:***

***continue***

***# If it's the same word, accumulate the count***

***if current\_word == word:***

***current\_count += count***

***else:***

***# If it's a new word, print the previous word and its count***

***if current\_word:***

***print(f'{current\_word}\t{current\_count}')***

***current\_word = word***

***current\_count = count***

***# Print the last word and its count***

***if current\_word == word:***

***print(f'{current\_word}\t{current\_count}')***

***if \_\_name\_\_ == "\_\_main\_\_":***

***reducer()***

* 1. Run the MapReduce job and display the output.

***Cat|input.txt|python mapper.py***

***Cat|input.txt|python mapper.py|sort|reducer.py***

1. Load a dataset and perform basic data transformations in Pig:
   1. Load the dataset product\_sales.csv with the schema {product\_id, product\_name, sales\_amount, category} into Pig.

***-- Load the dataset product\_sales.csv***

***product\_sales = LOAD 'product\_sales.csv'***

***USING PigStorage(',')***

***AS (product\_id:int, product\_name:chararray, sales\_amount:float, category:chararray);***

* 1. Filter products with a sales amount greater than 1000.

***-- Filter products with a sales amount greater than 1000***

***high\_sales\_products = FILTER product\_sales BY sales\_amount > 1000;***

* 1. Group the data by category.

***-- Group the data by category***

***grouped\_by\_category = GROUP high\_sales\_products BY category;***

* 1. Compute the total sales for each category.

***-- Compute the total sales for each category***

***category\_sales\_totals = FOREACH grouped\_by\_category***

***GENERATE group AS category, SUM(high\_sales\_products.sales\_amount) AS total\_sales;***

* 1. Store the result in a new dataset called category\_sales\_totals.

***-- Store the result in a new dataset called category\_sales\_totals***

***STORE category\_sales\_totals INTO 'category\_sales\_totals' USING PigStorage(',');***

## Sub. Code: P23DS3P6 | Sub. Name: Big Data Management and Analytics Lab | Semester: III

1. Use advanced Pig operations to perform grouping and filtering:
   1. Load the dataset customer\_transactions.csv with schema {transaction\_id, customer\_name, amount, category} into Pig.

***-- Load the dataset customer\_transactions.csv into Pig***

***customer\_transactions = LOAD 'customer\_transactions.csv'***

***USING PigStorage(',')***

***AS (transaction\_id:int, customer\_name:chararray, amount:float, category:chararray);***

* 1. Group the data by category.

***-- Group the data by category***

***grouped\_by\_category = GROUP customer\_transactions BY category;***

* 1. Calculate the total transaction amount for each category.

***-- Calculate the total transaction amount for each category***

***category\_total\_amount = FOREACH grouped\_by\_category***

***GENERATE group AS category, SUM(customer\_transactions.amount) AS total\_amount;***

* 1. Filter the transactions where the amount is greater than 1500.

***-- Filter transactions where the amount is greater than 1500***

***high\_value\_transactions = FILTER customer\_transactions BY amount > 1500;***

* 1. Store the result in a new dataset called high\_value\_transactions.

***-- Store the result in a new dataset called high\_value\_transactions***

***STORE high\_value\_transactions INTO 'high\_value\_transactions' USING PigStorage(',');***

1. Create a Hive table and query the data using basic SQL operations:
   1. Create a Hive table named departments with schema {dept\_id, dept\_name, budget}.

***-- Create the departments table in Hive***

***CREATE TABLE departments (***

***dept\_id INT,***

***dept\_name STRING,***

***budget FLOAT***

***);***

* 1. Load data from the file department\_budget.csv into the departments table.

***-- Load data from department\_budget.csv into the departments table***

***LOAD DATA INPATH '/path/to/department\_budget.csv' INTO TABLE departments;***

* 1. Write a query to find all departments with a budget greater than 1,000,000.

***-- Query to find all departments with a budget greater than 1,000,000***

***SELECT \* FROM departments***

***WHERE budget > 1000000;***

* 1. Write a query to calculate the average budget for all departments.

***-- Query to calculate the average budget for all departments***

***SELECT AVG(budget) AS avg\_budget FROM departments;***

* 1. Write a query to find the department with the smallest budget.

***-- Query to find the department with the smallest budget***

***SELECT \* FROM departments***

***ORDER BY budget ASC***

***LIMIT 1;***

## Sub. Code: P23DS3P6 | Sub. Name: Big Data Management and Analytics Lab | Semester: III

1. Implement a Word Count program using Hadoop MapReduce:
   1. Write a MapReduce program to count the occurrences of each word in the file reports.txt.
   2. Modify the program to exclude punctuation marks and special characters.

***#!/usr/bin/env python3***

***import sys***

***import re***

***def mapper():***

***# Regular expression to match words (ignore punctuation)***

***word\_pattern = re.compile(r'\b\w+\b')***

***for line in sys.stdin:***

***# Remove leading and trailing spaces***

***line = line.strip()***

***# Find all words in the line***

***words = word\_pattern.findall(line)***

***# Emit each word with a count of 1***

***for word in words:***

***print(f'{word.lower()}\t1')***

***if \_\_name\_\_ == "\_\_main\_\_":***

***mapper()***

***REDUCER():***

***#!/usr/bin/env python3***

***import sys***

***def reducer():***

***current\_word = None***

***current\_count = 0***

***word = None***

***# Read lines from the standard input (stdin)***

***for line in sys.stdin:***

***# Remove leading and trailing spaces***

***line = line.strip()***

***# Split the line into word and count***

***word, count = line.split('\t', 1)***

***try:***

***count = int(count)***

***except ValueError:***

***continue***

***# If this is the same word as the previous, increment the count***

***if current\_word == word:***

***current\_count += count***

***else:***

***if current\_word:***

***# Emit the word and its count***

***print(f'{current\_word}\t{current\_count}')***

***current\_word = word***

***current\_count = count***

***if current\_word == word:***

***# Emit the final word and count***

***print(f'{current\_word}\t{current\_count}')***

***if \_\_name\_\_ == "\_\_main\_\_":***

***reducer()***

* 1. Run the MapReduce job and display the output.

***cat|reports.txt|python mapper.py***

***cat|reports.txt|python mapper.py|sort|python reducer.py***

1. Load a dataset and perform basic transformations using Pig:
   1. Load the dataset sales\_performance.csv with the schema {salesperson\_id, sales\_amount, region} into Pig.

***-- Load the dataset sales\_performance.csv into Pig***

***sales\_data = LOAD 'sales\_performance.csv'***

***USING PigStorage(',')***

***AS (salesperson\_id:int, sales\_amount:float, region:chararray);***

* 1. Filter sales where the sales amount is greater than 2000.

***-- Filter sales where the sales amount is greater than 2000***

***high\_sales = FILTER sales\_data BY sales\_amount > 2000;***

* 1. Group the data by region.

***-- Group the data by region***

***grouped\_by\_region = GROUP high\_sales BY region;***

* 1. Calculate the average sales amount per region.

***-- Calculate the average sales amount per region***

***region\_sales\_avg = FOREACH grouped\_by\_region***

***GENERATE group AS region, AVG(high\_sales.sales\_amount) AS avg\_sales\_amount;***

* 1. Store the result in a new file region\_sales\_avg.

***-- Store the result in a new file region\_sales\_avg***

***STORE region\_sales\_avg INTO 'region\_sales\_avg' USING PigStorage(',');***

## Sub. Code: P23DS3P6 | Sub. Name: Big Data Management and Analytics Lab | Semester: III

1. Create and query tables in Hive using basic SQL operations:
   1. Create a Hive table named library\_books with schema {book\_id, book\_title, author, copies\_available}.

***CREATE TABLE library\_books (***

***book\_id INT,***

***book\_title STRING,***

***author STRING,***

***copies\_available INT***

***)***

***ROW FORMAT DELIMITED***

***FIELDS TERMINATED BY ','***

***STORED AS TEXTFILE;***

* 1. Load data from the file books\_data.csv into the Hive table.

***LOAD DATA INPATH '/path/to/books\_data.csv'***

***INTO TABLE library\_books;***

* 1. Write a query to find all books with fewer than 5 copies available.

***SELECT book\_title, author, copies\_available***

***FROM library\_books***

***WHERE copies\_available < 5;***

* 1. Write a query to calculate the total number of books in the library.

***SELECT SUM(copies\_available) AS total\_books***

***FROM library\_books;***

* 1. Write a query to find the author with the most books in the library.

***SELECT author, COUNT(\*) AS book\_count***

***FROM library\_books***

***GROUP BY author***

***ORDER BY book\_count DESC***

***LIMIT 1;***

1. Use advanced Pig operations to perform grouping and filtering:
   1. Load the dataset supplier\_data.csv with schema {supplier\_id, supplier\_name, rating, region} into Pig.

***-- Load the supplier data from supplier\_data.csv***

***suppliers = LOAD 'supplier\_data.csv'***

***USING PigStorage(',')***

***AS (supplier\_id:int, supplier\_name:chararray, rating:float, region:chararray);***

* 1. Group the data by region.

***-- Group the suppliers by region***

***grouped\_by\_region = GROUP suppliers BY region;***

* 1. Calculate the average rating of suppliers in each region.

***-- Calculate the average rating of suppliers in each region***

***avg\_rating\_per\_region = FOREACH grouped\_by\_region***

***GENERATE group AS region, AVG(suppliers.rating) AS avg\_rating;***

* 1. Filter suppliers with a rating greater than 4.5.

***-- Filter suppliers with a rating greater than 4.5***

***top\_suppliers = FILTER suppliers BY rating > 4.5;***

* 1. Store the result in a new dataset called top\_suppliers.

***-- Store the top suppliers in a new file***

***STORE top\_suppliers INTO 'top\_suppliers' USING PigStorage(',');***

## Sub. Code: P23DS3P6 | Sub. Name: Big Data Management and Analytics Lab | Semester: III

1. Load a dataset and perform basic data transformations in Pig:
   1. Load the dataset student\_grades.csv with the schema {student\_id, student\_name, subject, grade} into Pig.

***-- Load the student grades dataset***

***student\_grades = LOAD 'student\_grades.csv'***

***USING PigStorage(',')***

***AS (student\_id:int, student\_name:chararray, subject:chararray, grade:int);***

* 1. Filter students with grades higher than 90.

***-- Filter students with grades higher than 90***

***high\_achievers = FILTER student\_grades BY grade > 90;***

* 1. Group the data by subject.

***-- Group the filtered data by subject***

***grouped\_by\_subject = GROUP high\_achievers BY subject;***

* 1. Calculate the average grade for each subject.

***-- Calculate the average grade for each subject***

***avg\_grade\_per\_subject = FOREACH grouped\_by\_subject***

***GENERATE group AS subject, AVG(high\_achievers.grade) AS avg\_grade;***

* 1. Store the result in a new dataset called subject\_avg\_grades.

***-- Store the results in a new file named subject\_avg\_grades***

***STORE avg\_grade\_per\_subject INTO 'subject\_avg\_grades' USING PigStorage(',');***

1. Implement a Word Count program using Hadoop MapReduce:
   1. Write a MapReduce program to count the occurrences of each word in the file reviews.txt.
   2. Exclude stop words like "and", "of", "it" from the count.

***#!/usr/bin/env python3***

***import sys***

***import string***

***stop\_words = set(["and", "of", "it", "the", "a", "in", "to", "for", "is", "on", "that"])***

***def mapper():***

***for line in sys.stdin:***

***# Remove punctuation***

***line = line.translate(str.maketrans('', '', string.punctuation))***

***# Split the line into words***

***words = line.strip().split()***

***for word in words:***

***if word.lower() not in stop\_words:***

***print(f'{word}\t1')***

***if \_\_name\_\_ == "\_\_main\_\_":***

***mapper()***

***REDUCER():***

***#!/usr/bin/env python3***

***import sys***

***def reducer():***

***current\_word = None***

***current\_count = 0***

***for line in sys.stdin:***

***word, count = line.strip().split('\t', 1)***

***try:***

***count = int(count)***

***except ValueError:***

***continue***

***if current\_word == word:***

***current\_count += count***

***else:***

***if current\_word:***

***print(f'{current\_word}\t{current\_count}')***

***current\_word = word***

***current\_count = count***

***if current\_word == word:***

***print(f'{current\_word}\t{current\_count}')***

***if \_\_name\_\_ == "\_\_main\_\_":***

***reducer()***

* 1. Run the MapReduce job and display the output.

***Cat reviews.txt|python mapper.py***

***Cat reviews.txt | python mapper.py | sort | python reducer.py***

## Sub. Code: P23DS3P6 | Sub. Name: Big Data Management and Analytics Lab | Semester: III

1. Create a Hive table and query the data using basic SQL operations:
   1. Create a Hive table named hotel\_bookings with schema {booking\_id, customer\_name, check\_in, total\_cost}.

***CREATE TABLE hotel\_bookings (***

***booking\_id INT,***

***customer\_name STRING,***

***check\_in DATE,***

***total\_cost FLOAT***

***)***

***ROW FORMAT DELIMITED***

***FIELDS TERMINATED BY ','***

***STORED AS TEXTFILE;***

* 1. Load data from the file bookings.csv into the Hive table.

***LOAD DATA INPATH '/path/to/bookings.csv'***

***INTO TABLE hotel\_bookings;***

* 1. Write a query to find all bookings with a total cost greater than 5000.

***SELECT \* FROM hotel\_bookings***

***WHERE total\_cost > 5000;***

* 1. Write a query to calculate the total revenue from all bookings.

***SELECT SUM(total\_cost) AS total\_revenue***

***FROM hotel\_bookings;***

* 1. Write a query to find the booking with the highest total cost.

***SELECT \* FROM hotel\_bookings***

***ORDER BY total\_cost DESC***

***LIMIT 1;***

1. Load a dataset and perform basic transformations using Pig:
   1. Load the dataset order\_data.csv with the schema {order\_id, customer\_name, product, total\_price} into Pig.

***-- Load the order data***

***order\_data = LOAD 'order\_data.csv'***

***USING PigStorage(',')***

***AS (order\_id:int, customer\_name:chararray, product:chararray, total\_price:float);***

* 1. Filter orders where the total price is greater than 1200.

***-- Filter orders with total price greater than 1200***

***high\_price\_orders = FILTER order\_data BY total\_price > 1200;***

* 1. Group the data by product.

***-- Group the data by product***

***grouped\_by\_product = GROUP order\_data BY product;***

* 1. Calculate the average total price per product.

***-- Calculate the average total price per product***

***avg\_price\_per\_product = FOREACH grouped\_by\_product***

***GENERATE group AS product, AVG(order\_data.total\_price) AS avg\_total\_price;***

* 1. Store the result in a new file product\_avg\_price.

***-- Store the result in a new file***

***STORE avg\_price\_per\_product INTO 'product\_avg\_price' USING PigStorage(',');***

## Sub. Code: P23DS3P6 | Sub. Name: Big Data Management and Analytics Lab | Semester: III

1. Use advanced Pig operations to perform grouping and filtering:
   1. Load the dataset employee\_hours.csv with schema {emp\_id, emp\_name, hours\_worked, department} into Pig.

***-- Load the employee hours dataset***

***employee\_data = LOAD 'employee\_hours.csv'***

***USING PigStorage(',')***

***AS (emp\_id:int, emp\_name:chararray, hours\_worked:int, department:chararray);***

* 1. Group the data by department.

***-- Group data by department***

***grouped\_by\_department = GROUP employee\_data BY department;***

* 1. Calculate the total hours worked for each department.

***-- Calculate total hours worked per department***

***total\_hours\_per\_department = FOREACH grouped\_by\_department***

***GENERATE group AS department, SUM(employee\_data.hours\_worked) AS total\_hours;***

* 1. Filter employees who have worked more than 200 hours.

***-- Filter employees who have worked more than 200 hours***

***high\_hours\_employees = FILTER employee\_data BY hours\_worked > 200;***

* 1. Store the result in a new dataset called high\_hours\_employees.

***-- Store the result in a new dataset***

***STORE high\_hours\_employees INTO 'high\_hours\_employees' USING PigStorage(',');***

1. Implement a Word Count program using Hadoop MapReduce:
   1. Write a MapReduce program to count the occurrences of each word in the file forum\_posts.txt.
   2. Modify the program to exclude words shorter than 3 characters.

***import sys***

***import re***

***# Mapper function with filter for words longer than 2 characters***

***for line in sys.stdin:***

***words = re.findall(r'\w+', line.lower())***

***for word in words:***

***if len(word) >= 3:***

***print(f'{word}\t1')***

***reducer():***

***import sys***

***current\_word = None***

***current\_count = 0***

***word = None***

***# Reducer function***

***for line in sys.stdin:***

***word, count = line.split('\t')***

***count = int(count)***

***if current\_word == word:***

***current\_count += count***

***else:***

***if current\_word:***

***# Output the word and its count***

***print(f'{current\_word}\t{current\_count}')***

***current\_word = word***

***current\_count = count***

***# Output the last word***

***if current\_word == word:***

***print(f'{current\_word}\t{current\_count}')***

* 1. Run the MapReduce job and display the output.

***Cat forum\_posts.txt | python mapper.py***

***Cat forum\_posts.txt | python mapper.py |sort | python reducer.py***

## Sub. Code: P23DS3P6 | Sub. Name: Big Data Management and Analytics Lab | Semester: III

1. Load a dataset and perform basic data transformations in Pig:
   1. Load the dataset hospital\_patients.csv with the schema {patient\_id, patient\_name, age, diagnosis} into Pig.

***-- Load the hospital patients dataset***

***patients\_data = LOAD 'hospital\_patients.csv'***

***USING PigStorage(',')***

***AS (patient\_id:int, patient\_name:chararray, age:int, diagnosis:chararray);***

* 1. Filter patients with age greater than 60.

***-- Filter patients with age greater than 60***

***elderly\_patients = FILTER patients\_data BY age > 60;***

* 1. Group the data by diagnosis.

***-- Group the data by diagnosis***

***grouped\_by\_diagnosis = GROUP elderly\_patients BY diagnosis;***

* 1. Calculate the average age of patients for each diagnosis.

***-- Calculate the average age of patients for each diagnosis***

***avg\_age\_per\_diagnosis = FOREACH grouped\_by\_diagnosis***

***GENERATE group AS diagnosis, AVG(elderly\_patients.age) AS avg\_age;***

* 1. Store the result in a new dataset called diagnosis\_avg\_age.

***-- Store the result in a new dataset***

***STORE avg\_age\_per\_diagnosis INTO 'diagnosis\_avg\_age' USING PigStorage(',');***

1. Create a Hive table and query the data using basic SQL operations:
   1. Create a Hive table named movie\_ratings with schema {movie\_id, movie\_name, rating}.

***-- Create the movie\_ratings table***

***CREATE TABLE movie\_ratings (***

***movie\_id INT,***

***movie\_name STRING,***

***rating FLOAT***

***)***

***ROW FORMAT DELIMITED***

***FIELDS TERMINATED BY ',';***

* 1. Load data from the file ratings.csv into the Hive table.

***-- Load data from the ratings.csv file into the movie\_ratings table***

***LOAD DATA INPATH '/path/to/ratings.csv' INTO TABLE movie\_ratings;***

* 1. Write a query to find all movies with ratings greater than 4.5.

***-- Query to find all movies with ratings greater than 4.5***

***SELECT movie\_name, rating***

***FROM movie\_ratings***

***WHERE rating > 4.5;***

* 1. Write a query to calculate the average rating of all movies.

***-- Query to calculate the average rating of all movies***

***SELECT AVG(rating) AS avg\_rating***

***FROM movie\_ratings;***

* 1. Write a query to find the highest-rated movie.

***-- Query to find the highest-rated movie***

***SELECT movie\_name, rating***

***FROM movie\_ratings***

***ORDER BY rating DESC***

***LIMIT 1;***

## Sub. Code: P23DS3P6 | Sub. Name: Big Data Management and Analytics Lab | Semester: III

1. Implement a Word Count program using Hadoop MapReduce:
   1. Write a MapReduce program to count the occurrences of each word in the file news\_articles.txt.
   2. Modify the program to exclude all numeric digits from the count.

***import sys***

***import re***

***# Mapper function***

***def mapper():***

***for line in sys.stdin:***

***# Remove leading and trailing whitespaces***

***line = line.strip()***

***# Split the line into words using regular expression***

***words = re.split(r'\W+', line.lower())***

***# Emit words without numeric digits***

***for word in words:***

***if word and not any(char.isdigit() for char in word):***

***print(f"{word}\t1")***

***if \_\_name\_\_ == "\_\_main\_\_":***

***mapper()***

***REDUCER()***

***import sys***

***# Reducer function***

***def reducer():***

***current\_word = None***

***current\_count = 0***

***word = None***

***# Input comes from STDIN***

***for line in sys.stdin:***

***# Remove leading and trailing whitespaces***

***line = line.strip()***

***# Parse the input from mapper.py (word \t count)***

***word, count = line.split('\t', 1)***

***# Convert count from string to int***

***try:***

***count = int(count)***

***except ValueError:***

***continue***

***if current\_word == word:***

***current\_count += count***

***else:***

***if current\_word:***

***print(f"{current\_word}\t{current\_count}")***

***current\_word = word***

***current\_count = count***

***if current\_word == word:***

***print(f"{current\_word}\t{current\_count}")***

***if \_\_name\_\_ == "\_\_main\_\_":***

***reducer()***

* 1. Run the MapReduce job and display the output.

***Cat newsarticle.txt | python mapper.py***

***Cat newsaryicle.txt | python mapper.py |sort | python reducer.py***

1. Load a dataset and perform basic transformations using Pig:
   1. Load the dataset employee\_projects.csv with the schema {emp\_id, emp\_name, project, hours\_spent} into Pig.

***-- Load the employee projects dataset***

***employee\_data = LOAD 'employee\_projects.csv'***

***USING PigStorage(',')***

***AS (emp\_id:int, emp\_name:chararray, project:chararray, hours\_spent:int);***

* 1. Filter employees who spent more than 100 hours on projects.

***-- Filter employees who spent more than 100 hours***

***over\_100\_hours = FILTER employee\_data BY hours\_spent > 100;***

* 1. Group the data by project.

***-- Group data by project***

***grouped\_by\_project = GROUP over\_100\_hours BY project;***

* 1. Calculate the total hours spent on each project.

***-- Calculate the total hours spent on each project***

***total\_hours\_per\_project = FOREACH grouped\_by\_project***

***GENERATE group AS project, SUM(over\_100\_hours.hours\_spent) AS total\_hours;***

* 1. Store the result in a new dataset called project\_total\_hours.

***-- Store the result in a new dataset***

***STORE total\_hours\_per\_project INTO 'project\_total\_hours' USING PigStorage(',');***

## Sub. Code: P23DS3P6 | Sub. Name: Big Data Management and Analytics Lab | Semester: III

1. Load a dataset and perform basic transformations using Pig:
   1. Load the dataset gym\_members.csv with schema {member\_id, member\_name, age, membership\_type} into Pig.

***-- Load the gym members dataset***

***gym\_members = LOAD 'gym\_members.csv'***

***USING PigStorage(',')***

***AS (member\_id:int, member\_name:chararray, age:int, membership\_type:chararray);***

* 1. Filter members with a "premium" membership type.

***-- Filter members with premium membership***

***premium\_members = FILTER gym\_members BY membership\_type == 'premium';***

* 1. Group the data by age group (e.g., 20-30, 30-40).

***-- Create age group***

***age\_group\_members = FOREACH gym\_members GENERATE***

***(age >= 20 AND age < 30 ? '20-30' :***

***(age >= 30 AND age < 40 ? '30-40' : '40+')) AS age\_group,***

***member\_id, member\_name, age, membership\_type;***

***-- Group the data by age group***

***grouped\_by\_age = GROUP age\_group\_members BY age\_group;***

* 1. Calculate the total number of members in each age group.

***-- Calculate the total number of members in each age group***

***total\_members\_per\_age\_group = FOREACH grouped\_by\_age***

***GENERATE group AS age\_group, COUNT(age\_group\_members) AS total\_members;***

* 1. Store the result in a new dataset called age\_group\_members.

***-- Store the result in a new dataset***

***STORE total\_members\_per\_age\_group INTO 'age\_group\_members' USING PigStorage(',');***

1. Create and query tables in Hive using basic SQL operations:
   1. Create a Hive table named car\_sales with schema {car\_id, model, price, year}.

***CREATE TABLE car\_sales (***

***car\_id INT,***

***model STRING,***

***price FLOAT,***

***year INT***

***)***

***ROW FORMAT DELIMITED***

***FIELDS TERMINATED BY ','***

***STORED AS TEXTFILE;***

* 1. Load data from the file car\_sales.csv into the Hive table.

***LOAD DATA INPATH '/path/to/car\_sales.csv' INTO TABLE car\_sales;***

* 1. Write a query to find all cars sold after the year 2020.

***SELECT \****

***FROM car\_sales***

***WHERE year > 2020;***

* 1. Write a query to calculate the total sales value of all cars.

***SELECT SUM(price) AS total\_sales\_value***

***FROM car\_sales;***

* 1. Write a query to find the car with the highest price.

***SELECT \****

***FROM car\_sales***

***ORDER BY price DESC***

***LIMIT 1;***

## Sub. Code: P23DS3P6 | Sub. Name: Big Data Management and Analytics Lab | Semester: III

1. Create a Hive table and query the data using basic SQL operations:
   1. Create a Hive table named holiday\_bookings with schema {booking\_id, destination, total\_cost}.

***CREATE TABLE holiday\_bookings (***

***booking\_id INT,***

***destination STRING,***

***total\_cost FLOAT***

***) ROW FORMAT DELIMITED***

***FIELDS TERMINATED BY ','***

***STORED AS TEXTFILE;***

* 1. Load data from the file holiday\_bookings.csv into the Hive table.

***LOAD DATA INPATH '/path/to/holiday\_bookings.csv' INTO TABLE holiday\_bookings;***

* 1. Write a query to find all bookings with a total cost greater than 3000.

***SELECT \****

***FROM holiday\_bookings***

***WHERE total\_cost > 3000;***

* 1. Write a query to calculate the average total cost of all bookings.

***SELECT AVG(total\_cost) AS average\_total\_cost***

***FROM holiday\_bookings;***

* 1. Write a query to find the booking with the lowest total cost.

***SELECT \****

***FROM holiday\_bookings***

***ORDER BY total\_cost ASC***

***LIMIT 1;***

1. Use advanced Pig operations to perform grouping and filtering:
   1. Load the dataset sales\_team.csv with schema {emp\_id, emp\_name, target, region} into Pig.

***-- Load the sales team dataset***

***sales\_team = LOAD 'sales\_team.csv'***

***USING PigStorage(',')***

***AS (emp\_id:int, emp\_name:chararray, target:int, region:chararray);***

* 1. Group the data by region.

***-- Group the data by region***

***grouped\_by\_region = GROUP sales\_team BY region;***

* 1. Calculate the total target achieved by each region.

***-- Calculate total target per region***

***total\_target\_per\_region = FOREACH grouped\_by\_region***

***GENERATE group AS region, SUM(sales\_team.target) AS total\_target;***

* 1. Filter employees with a target greater than 800.

***-- Filter employees with target greater than 800***

***top\_employees = FILTER sales\_team BY target > 800;***

* 1. Store the result in a new dataset called top\_sales\_team.

***-- Store the result in a new file called top\_sales\_team***

***STORE top\_employees INTO 'top\_sales\_team' USING PigStorage(',');***