Set1:

1. Implement a Word Count program using Hadoop MapReduce:
   a) 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()
```

   b) Run the MapReduce job and display the output.

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

2. Load a dataset and perform basic transformations using Pig:

a) 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);

b) Filter employees with a salary greater than 80,000.

high_salary_employees = FILTER employee_data BY salary > 80000;

c) Group the data by department.

grouped_by_department = GROUP employee_data BY department;

d) 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;

e) 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:

a) 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);

b) 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;

c) Group the data by region.

-- Group the data by region
grouped_by_region = GROUP high_value_orders BY region;

d) 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;

e) 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(',');

2. Use advanced Pig operations to perform grouping and filtering:

a) 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);

b) Group the data by category.

-- Group the data by category

grouped_by_category = GROUP product_data BY category;

c) 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;

d) 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;

e) 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:

a) 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;
```

b) Load data from the file employees.csv into the employees table.

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

c) Write a query to select all employees from the "HR" department.

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

d) Write a query to find employees with a salary greater than 70,000.

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

e) Write a query to calculate the average salary by department.

```
SELECT department, AVG(salary) AS average_salary
FROM employees
    GROUP BY department;
```

2. Implement a Word Count program using Hadoop MapReduce:

a) 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()
```

b) 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:

   a) 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);

b) 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;

c) Group the data by department.

-- Group the data by department

grouped_by_department = GROUP experienced_employees BY department;

d) 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;

e) 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(',');

2. Create and query tables in Hive using basic SQL operations:

a) 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;
```

b) Load data from the file inventory_data.csv into the Hive table.

LOAD DATA INPATH 'inventory_data.csv' INTO TABLE inventory;

c) Write a query to find all items with quantity less than 50.

SELECT * FROM inventory WHERE quantity < 50;

d) 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;

e) 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:

a) 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);

b) Group the data by region.

-- Group the data by region
grouped_by_region = GROUP sales_data BY region;

c) 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;

d) Filter the sales where the sale amount is greater than 500.

filtered_sales = FILTER sales_data BY sale_amount > 500;

e) Store the result in a new dataset called region_sales_data.

STORE filtered_sales INTO 'region_sales_data' USING PigStorage(',');

2. Create a Hive table and query the data using basic SQL operations:

a) 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;
```

b) Load data from the file students.csv into the students table.

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

c) Write a query to select all students with grades greater than 85.

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

d) Write a query to calculate the average grade for each subject.

```
SELECT subject, AVG(grade) AS average_grade
FROM students
    GROUP BY subject;
```

e) 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.

b) 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()
```

c) Run the MapReduce job and display the output.

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

2. Load a dataset and perform basic data transformations in Pig:

a) 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);
```

b) Filter records where the age is greater than 40.

```
-- Filter customers with age greater than 40
filtered_customers = FILTER customer_data BY age > 40;
```

c) Group the data by country.

```
-- Group the data by country
grouped_by_country = GROUP filtered_customers BY country;
```

d) 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;
```

e) 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:

a) 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);

b) 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;

c) Group the data by transaction_date.

-- Group transactions by transaction_date
grouped_by_date = GROUP high_value_transactions BY transaction_date;

d) 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;

e) 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(',');

2. Create and query tables in Hive using basic SQL operations:

a) 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 ',';
```

b) Load data from the file courses.csv into the Hive table.

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

c) Write a query to find all courses with a duration less than 6 months

```
SELECT * FROM courses
WHERE duration < 6;
```

d) Write a query to calculate the total fee for all courses.

```
SELECT SUM(fee) AS total_fee FROM courses;
```

e) 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()
```

c) Run the MapReduce job and display the output.

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

2. Load a dataset and perform basic data transformations in Pig:

a) 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);
```

b) 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;
```

c) Group the data by category.

```
-- Group the data by category
grouped_by_category = GROUP high_sales_products BY category;
```

d) 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;
```

e) 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:

a) Load the dataset customer_transactions.csvwith 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);

b) Group the data by category.

```
-- Group the data by category
grouped_by_category = GROUP customer_transactions BY category;
```

c) 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;
```

d) 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;
```

e) 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(',');
```

2. Create a Hive table and query the data using basic SQL operations:

a) Create a Hive table named departmentswith schema {dept_id, dept_name, budget}.

```
-- Create the departments table in Hive
CREATE TABLE departments (
    dept_id INT,
    dept_name STRING,
    budget FLOAT
);
```

b) 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;
```

c) 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;
```

d) 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;
```

e) 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:

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

b) 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()
```

c) Run the MapReduce job and display the output.

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

2. Load a dataset and perform basic transformations using Pig:

a) Load the dataset sales_performance.csvwith the schema {salesperson_id, sales_amount, region} intoPig.

```
-- 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);
```

b) 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;
```

c) Group the data by region.

```
-- Group the data by region
grouped_by_region = GROUP high_sales BY region;
```

d) 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;
```

e) 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:

a) Create a Hive table named library_bookswith 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;
```

b) Load data from the file books_data.csvinto the Hive table.

```
LOAD DATA  INPATH '/path/to/books_data.csv'
INTO TABLE library_books;
```

c) 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;
```

d) Write a query to calculate the total number of books in the library.

```
SELECT SUM(copies_available) AS total_books
FROM library_books;
```

e) 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;
```

2. Use advanced Pig operations to perform grouping and filtering:

a) Load the dataset supplier_data.csvwith 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);
```

b) Group the data by region.

```
-- Group the suppliers by region
grouped_by_region = GROUP suppliers BY region;
```

c) 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;
```

d) 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;
```

e) 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:

a) Load the dataset student_grades.csv with the schema {student_id, student_name, subject, grade} intoPig.

```
-- Load the student grades dataset
student_grades = LOAD 'student_grades.csv'
USING PigStorage(',')
   AS (student_id:int, student_name:chararray, subject:chararray, grade:int);
```

b) Filter students with grades higher than 90.

```
-- Filter students with grades higher than 90
high_achievers = FILTER student_grades BY grade > 90;
```

c) Group the data by subject.

```
-- Group the filtered data by subject
grouped_by_subject = GROUP high_achievers BY subject;
```

d) 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;
```

e) 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(',');
```

2. Implement a Word Count program using Hadoop MapReduce:

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

b) 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()
```

c) 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:

   a) Create a Hive table named hotel_bookingswith 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;

b) Load data from the file bookings.csv into the Hive table.

LOAD DATA INPATH '/path/to/bookings.csv'
INTO TABLE hotel_bookings;

c) Write a query to find all bookings with a total cost greater than 5000.

SELECT * FROM hotel_bookings
WHERE total_cost > 5000;

d) Write a query to calculate the total revenue from all bookings.

SELECT SUM(total_cost) AS total_revenue
FROM hotel_bookings;

e) Write a query to find the booking with the highest total cost.

SELECT * FROM hotel_bookings
ORDER BY total_cost DESC
LIMIT 1;

2. Load a dataset and perform basic transformations using Pig:

a) 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);

b) 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;

c) Group the data by product.

-- Group the data by product
grouped_by_product = GROUP order_data BY product;

d) 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;

e) 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:

a) Load the dataset employee_hours.csvwith 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);

b) Group the data by department.

-- Group data by department
grouped_by_department = GROUP employee_data BY department;

c) 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;

d) 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;

e) 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(',');

2. Implement a Word Count program using Hadoop MapReduce:

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

b) 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}')

c) 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:

a) 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);

b) Filter patients with age greater than 60.

-- Filter patients with age greater than 60

elderly_patients = FILTER patients_data BY age > 60;

c) Group the data by diagnosis.

-- Group the data by diagnosis

grouped_by_diagnosis = GROUP elderly_patients BY diagnosis;

d) 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;

e) 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(',');

2. Create a Hive table and query the data using basic SQL operations:

a) Create a Hive table named movie_ratingswith 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 ',';

b) Load data from the file ratings.csvinto 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;

c) 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;
```

d) 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;
```

e) 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:

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

b) 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()
```

c) Run the MapReduce job and display the output.

```
Cat newsarticle.txt | python mapper.py
Cat newsaryicle.txt | python mapper.py |sort | python reducer.py
```

2. Load a dataset and perform basic transformations using Pig:

a) Load the dataset employee_projects.csvwith 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);
```

b) 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;
```

c) Group the data by project.

```
-- Group data by project
grouped_by_project = GROUP over_100_hours BY project;
```

d) 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;

e) 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:

a) Load the dataset gym_members.csvwith 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);
```

b) Filter members with a "premium" membership type.

```
-- Filter members with premium membership
premium_members = FILTER gym_members BY membership_type == 'premium';
```

c) 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;
```

d) 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;
```

e) 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(',');
```

2. Create and query tables in Hive using basic SQL operations:

a) Create a Hive table named car_saleswith 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;
```

b) Load data from the file car_sales.csvinto the Hive table.

```
LOAD DATA INPATH '/path/to/car_sales.csv' INTO TABLE car_sales;
```

c) Write a query to find all cars sold after the year 2020.

```
SELECT *
FROM car_sales
   WHERE year > 2020;
```

d) Write a query to calculate the total sales value of all cars.

```
SELECT SUM(price) AS total_sales_value
```

FROM car_sales;

e) 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:

a) 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;
```

b) Load data from the file holiday_bookings.csv into the Hive table.

```
LOAD DATA INPATH '/path/to/holiday_bookings.csv' INTO TABLE holiday_bookings;
```

c) Write a query to find all bookings with a total cost greater than 3000.

```
SELECT *
FROM holiday_bookings
    WHERE total_cost > 3000;
```

d) Write a query to calculate the average total cost of all bookings.

```
SELECT AVG(total_cost) AS average_total_cost
FROM holiday_bookings;
```

e) Write a query to find the booking with the lowest total cost.

```
SELECT *
FROM holiday_bookings
ORDER BY total_cost ASC
LIMIT 1;
```

2. Use advanced Pig operations to perform grouping and filtering:

a) Load the dataset sales_team.csvwith 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);
```

b) Group the data by region.

```
-- Group the data by region
grouped_by_region = GROUP sales_team BY region;
```

c) 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;
```

d) Filter employees with a target greater than 800.

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

top_employees = FILTER sales_team BY target > 800;

e) 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(',');