Big Data Analytics Lab

PySpark

Ex - 5(i) - PySpark WordCount

wordCount.py

```
from pyspark import SparkContext

sc = SparkContext(appName="WordCount")
input_file = "inputFile.txt"
lines = sc.textFile(input_file)

word_counts = lines.flatMap(lambda line : line.split())
    .map(lambda word : (word,1)).reduceByKey(lambda a, b: a + b)

for word, count in word_counts.collect():
        print(f"{word}\t{count}")

sc.stop()
```

shell command

```
~/BigData/spark-3.5.3-bin-hadoop3/bin/spark-submit wordCount.py
# navigate to directory where spark-submit is present - spark-3....
# this command to be typed in /BigData/ex5_1 not in spark-3....
```

Ex - 5(ii) - PySpark WordCount

inputFile.txt - MovieRatings Dataset

movie_ratings.py

```
from pyspark import SparkContext

sc = SparkContext("local", "Movie Ratings Distribution")

text_file = sc.textFile("inputFile.txt")

movie_ratings = text_file.map(lambda line: line.split("\t")).map(lambda fields: (fields[1],
int(fields[2])))

ratings_distribution = movie_ratings.map(lambda x : ((x[0], x[1]), 1)).reduceByKey(lambda a, b
: a + b).map(lambda x: (x[0][0], (x[0][1], x[1]))).groupByKey().mapValues(list)
```

```
output = ratings_distribution.collect()
for (movie_id, ratings) in output:
    print(f"Movie ID:{movie_id}, Ratings Distribution:{ratings}")

sc.stop()
```

Ex - 6

1. Use the “friends_test” dataset. Col1 is ID, Col2 is name, Col 3 is Age, Col 4 is num of friends. Understand **mapvalues function of RDD** in spark and find the average number of friends for each unique age present in the dataset.

2. Use the “temp.csv” dataset. Column headers are present in the dataset. Understand filter operations and filter out only the “TMIN” values from the “desc” column. With the resultant data (RDD) find the following:

    a. Minimum temperature (overall)
    b. Minimum temperature for every ItemID
    c. Minimum temperature for every StationID.

3. Use the same dataset, filter only “TMAX” column and find the maximum temperatures just like the ones mentioned above.

Ex - 6(i) - Mapvalue function of RDD

num_frnds.py

```
from pyspark import SparkContext

sc = SparkContext("local", "Friends Average")

data = sc.textFile("friends_test.csv").map(lambda line: line.split(","))

age_friends_rdd = data.map(lambda x: (int(x[2]), int(x[3])))

combined = age_friends_rdd.combineByKey(
    lambda friends: (friends, 1),
    lambda acc, friends: (acc[0] + friends, acc[1] + 1),
    lambda acc1, acc2: (acc1[0] + acc2[0], acc1[1] + acc2[1])
)

average_friends_rdd = combined.mapValues(lambda x: x[0] / x[1])
```

```
results = average_friends_rdd.collect()
for age, avg_friends in results:
    print(f"Age: {age}, Average Number of Friends: {avg_friends}")

sc.stop()
```

Ex - 6(ii)

Data - temp.csv

temp.py

```
from pyspark import SparkContext

sc = SparkContext("local", "Temperature Analysis")

data = sc.textFile("temp.csv").zipWithIndex().filter(lambda x: x[1] != 0).map(lambda x:
x[0].split(","))

temp_rdd = data.map(lambda x: (x[0], x[1], x[2], int(x[3])))
tmin_rdd = temp_rdd.filter(lambda x: x[2] == "TMIN")

overall_min_temp = tmin_rdd.map(lambda x: x[3]).min()
print(f"Overall Minimum Temperature: {overall_min_temp}")

min_temp_per_itemID = tmin_rdd.map(lambda x: (x[0], x[3])).reduceByKey(lambda a, b: min(a, b))
print("Minimum Temperature for each ItemID:")
for itemID, min_temp in min_temp_per_itemID.collect():
    print(f"ItemID: {itemID}, Minimum Temperature: {min_temp}")

min_temp_per_stationID = tmin_rdd.map(lambda x: (x[1], x[3])).reduceByKey(lambda a, b: min(a,
b))
print("Minimum Temperature for each StationID:")
for stationID, min_temp in min_temp_per_stationID.collect():
    print(f"StationID:{stationID}, Minimum Temperature:{min_temp}")

sc.stop()
```

Ex - 6(iii)

```
from pyspark import SparkContext

sc = SparkContext("local", "Temperature Analysis")

data = sc.textFile("temp.csv").zipWithIndex().filter(lambda x: x[1] != 0).map(lambda x:
x[0].split(","))
temp_rdd = data.map(lambda x: (x[0], x[1], x[2], int(x[3])))
tmax_rdd = temp_rdd.filter(lambda x: x[2] == "TMAX")
```

```
overall_max_temp = tmax_rdd.map(lambda x: x[3]).max()
print(f"Overall Maximum Temperature: {overall_max_temp}")

max_temp_per_itemID = tmax_rdd.map(lambda x: (x[0], x[3])).reduceByKey(lambda a, b: max(a, b))
print("Maximum Temperature for each ItemID:")
for itemID, max_temp in max_temp_per_itemID.collect():
    print(f"ItemID: {itemID}, Maximum Temperature: {max_temp}")

max_temp_per_stationID = tmax_rdd.map(lambda x: (x[1], x[3])).reduceByKey(lambda a, b: max(a,
b))
print("Maximum Temperature for each StationID:")
for stationID, max_temp in max_temp_per_stationID.collect():
    print(f"StationID: {stationID}, Maximum Temperature: {max_temp}")

sc.stop()
```

AWS

Ex - 1 - EC2

1. **Login to AWS Console**
   - Access your AWS account.
2. **Select VPC Service**
   - Choose the **VPC** service.
   - Ensure you are in the **Mumbai** region.
3. **Delete Existing VPC**
   - Remove any pre-existing VPCs to start with a clean setup.
4. **Create a Custom VPC and Components**
   - Create a new VPC with the following settings:
     - **VPC Name**: `snu-vpc`
     - **CIDR**: `192.168.0.0/16`
   - Create a **public subnet**:
     - **Subnet Name**: `public-subnet`
     - **CIDR**: `192.168.1.0/24`

- Set up an **Internet Gateway**:
  - **IGW Name**: `snu-igw`
  - Attach the Internet Gateway to `snu-vpc`.

5. **Configure the Route Table**
   - Go to **Route Tables**.
   - Click on the Route Table ID associated with `snu-vpc`.
   - Edit the routes:
     - **Add route**: Destination `0.0.0.0/0` with target `snu-igw`.
6. **Allocate Elastic IPs**
   - Go to **Elastic IPs**.
   - Allocate two elastic IP addresses.
7. **Launch EC2 Instances**
   - Go to **EC2** service and select **Launch Instance**.
   - Configure two instances:
     - **Number of Instances**: 2
     - **Instance Type**: Select as required
     - **Key Pair**: No key pair (for practice environment)
   - Name the instances:
     - **Instance 1**: Web Server
     - **Instance 2**: Web Client
8. **Attach Elastic IPs to Instances**
   - Associate the previously allocated elastic IPs:
     - Go to **Elastic IPs**, select an IP, and associate it with **Web Server**.
     - Repeat for the **Web Client** instance.
9. **Connect to EC2 Instances**

- Connect to **Web Server** and **Web Client** individually via **EC2 Instance Connect**.

10. **Install Apache on Web Server**
    - On the Web Server, run:

```
ping 8.8.8.8
sudo apt update
sudo apt install apache2 -y
sudo service apache2 status
```

    - Verify Apache service is running.
11. **Install Links on Web Client**
    - On the Web Client, run:

```
ping 8.8.8.8
sudo apt update
sudo apt install links -y
```

12. **Configure Security Group for HTTP Access**
    - Go to **Security Groups** of the **Web Server**.
    - Edit inbound rules:
        - Add an HTTP rule:
            - **Type**: HTTP
            - **Source**: Anywhere (0.0.0.0/0)
13. **Modify Network ACL to Allow SSH and HTTP**
    - Check **Network ACLs** associated with `snu-vpc`.
    - Edit the rules:
        - Add rule:
            - **Rule number**: 100

- **Type**: All Traffic
- **Source**: 0.0.0.0/0
- **Action**: Allow

14. **Test Web Access from Web Client**
    - On the Web Client, run:

      ```
      links http://<Web Server IP>
      ```

    - Replace `<Web Server IP>` with the elastic IP address of the Web Server.

Ex 2 - Testing Route 53 Service with Custom Domain

1. **Create a Hosted Zone in Route 53**
    - Go to **Route 53** in the AWS Console.
    - Create a new **hosted zone** with the domain name:
        - **Domain**: `21011101122.ngaws.xyz`
    - AWS will automatically generate Name Servers (NS) for this hosted zone.
2. **Login to GoDaddy**
    - Go to https://www.godaddy.com/.
    - Login with the credentials:
        - **Username**: `aws-ng`
        - **Password**: `Welcome1!`
3. **Update Name Server Records in GoDaddy**
    - In GoDaddy, navigate to **DNS Management** for `21011101122.ngaws.xyz`.
    - Add a new **NS (Name Server) record** with the following:
        - **Name**: `21011101122`
        - **Type**: NS

- **Value**: Paste the name server information from Route 53's hosted zone.

4. **Create a Record in Route 53**
   - Go back to **Route 53** and open the hosted zone `21011101122.ngaws.xyz`.
   - Create a new record:
     - **Name**: `www`
     - **Type**: A (IPv4 Address)
     - **Value**: Enter the **IP address of your Web Server instance**.
     - **Routing Policy**: Simple
5. **Test Domain Reachability**
   - From the **Web Client** instance, verify domain reachability:
     - Test with the `links` browser:

```
links www.21011101122.ngaws.xyz
```

     - Run an **nslookup** command to check the DNS resolution:

```
nslookup www.21011101122.ngaws.xyz
```

Ex - 3 - Setting Up IAM Users with Console Access and Permissions

1. **Access IAM Service**
   - Search for **IAM** in the AWS Console and open it.
2. **Create an IAM User**
   - Go to **Users** in the IAM dashboard.
   - Click **Create user**.
   - Enter a **username** (e.g., `example_user`).

- Enable **AWS Management Console access**.
- Set the console password or choose an auto-generated one.
- Click **Next** and complete the user creation process.

3. **Copy User Sign-In Details**
   - After creating the user, copy the sign-in link, username, and password.
   - Use these details to log in as the new user in a separate tab.
4. **Assign Permissions to the User**
   - In the root account, go to **IAM**, then **Users**, and select the newly created user.
   - Click on **Add permissions**.
     - Choose **Attach policies directly** and select **EC2 Full Access** policy.
   - Alternatively, create a **User Group** with permissions:
     - Go to **User groups** and click **Create group**.
     - Enter a **Group name** (e.g., `EC2_Admins`).
     - Attach the **EC2 Full Access** policy.
     - Add the user to the group by selecting the user from the list.
5. **Create Another IAM User**
   - Repeat the process to create another IAM user with similar or different permissions as needed.

Creating Custom VPC, EC2 Instance and working on SG & NACL

1. Login into your AWS account.
2. Choose VPC Service
3. Choose the region Mumbai
4. Delete the existing VPC

5. Setup custom VPC and its components - Create VPC (snu-vpc: 192.168.0.0/16, public-subnet: 192.168.1.0/24, snu-igw: Attach to VPC)
6. Route Table - Click on Route Table ID, Routes - Edit Routes - Add Routes - 0.0.0.0/0 - internet gateway
7. Get 2 elastic public IP - attach elastic IP
8. Create two EC2 instances(search for ec2) - instances, launch instance (2, VMs, quick start, key pair name: no key pair name)
9. EC2 Instances - Name VM1 as Web Server & VM2 as Web Client
10. Attach the public IP address - Elastic IP Addresses, click on IP, associate IP, web server/client instance
11. Instances - Server/Client(one by one) - Connect to the instance via EC2 instance connect
12. Install Apache (web service) in Web Server - ping 8.8.8.8, sudo apt update, sudo apt install apache, sudo apt install apache2, service apache2 status
13. Install Links (web client) in Web Client - ping 8.8.8.8, sudo apt update, sudo apt install links
14. In the Security Group of Web Server, add rule to allow HTTP access : Instances - Click Instance ID of Server - Security - Security Groups - Edit inbound rules - add rules - HTTP, Anywhere, 0.0.0.0/0
15. Allow SSH & HTTP on the NACL - Instances, Web Server, Security - Check Network ACLs - Click on ID, Edit, Remove, Add - 100, all traffic, 0.0.0.0/0, Allow
16. Test the web access from the web client using links app - copy IP of web server and links <IP> on client, links http://<IP>

Testing Route53 Service

1. Create a hosted zone in AWS Route 53 service(search) – 21011101122.ngaws.xyz
2. Login to https://www.godaddy.com/, aws-ng – Welcome1!

3. Get the name server information from Route 53 dashboard(Value/Route traffic to) and update NS record in GoDaddy portal – add new record, NS, 21011101122, paste the server info
4. AWS Route 53 – Create a record in hosted zone – www, IP address of web server(take it from instances), simple
5. Check reachability - Web client – links www.21011101122.ngaws.xyz, nslookup www.21011101122.ngaws.xyz

IAM Service(Search)

Users - Create user, user_name, give console access, i want to create IAM user

Copy user details - sign in using these details in separate tab

Root account - IAM, click on the user, add permissions, attach policy(ec2 full access) or User Groups, create user group, user group name, select users, attach permissions

create another user

ESE

Question 1 -

Here's how to perform each step to create a VPC with two subnets, configure network components, set up EC2 instances, and install the necessary applications for a client-server setup in AWS.

1. Log in to AWS Console

- Go to the AWS Management Console and sign in.

2. Create a VPC (Virtual Private Cloud)

- Navigate to **VPC** service in the AWS Console.
- **Create VPC**:
  - **VPC Name**: `SNUC_VPC`

- **CIDR Block**: `192.168.0.0/16`
- Click **Create VPC**.

3. Create Subnets

- In the **Subnets** section, create two subnets:
  - **Subnet 1 (Server Subnet)**:
    - **Name**: `Server`
    - **VPC**: `SNUC_VPC`
    - **CIDR Block**: `192.168.1.0/24`
    - **Availability Zone**: Select any (e.g., `ap-south-1a` for Mumbai region).
  - **Subnet 2 (Client Subnet)**:
    - **Name**: `Client`
    - **VPC**: `SNUC_VPC`
    - **CIDR Block**: `192.168.2.0/24`
    - **Availability Zone**: Select any (e.g., `ap-south-1b`).

4. Set Up Internet Gateway

- Go to **Internet Gateways**.
- Click on **Create Internet Gateway**:
  - **Name**: `SNUC_IGW`
- After creation, **Attach** this internet gateway to `SNUC_VPC`.

5. Configure Route Table for Internet Access

- In **Route Tables**, locate the route table associated with `SNUC_VPC`.
- **Edit Routes** and add the following:
  - **Destination**: `0.0.0.0/0`
  - **Target**: Select `SNUC_IGW`
- Associate this route table with the `Server` subnet to allow internet access.

6. Create EC2 Instance: Web Server

- Go to **EC2** service and click on **Launch Instance**.
- Configure the instance as follows:
  - **Name**: `Web Server`
  - **Subnet**: Choose `Server` subnet.
  - **Auto-assign Public IP**: Enable.
- **Security Group**:
  - Create a new security group with the following inbound rule:
    - **Type**: HTTP, **Port**: 80, **Source**: Anywhere (0.0.0.0/0).
- Launch the instance.

7. Allocate and Attach Elastic IP

- Go to **Elastic IPs**.
- Allocate a new Elastic IP and associate it with the `Web Server` instance.

8. Connect to Web Server and Install Apache

- Use **EC2 Instance Connect** to open the CLI for `Web Server`.
- Update and install Apache:

```
sudo apt update
sudo apt install apache2 -y
sudo service apache2 start
```

- Confirm Apache is running with `sudo service apache2 status`.

9. Create Network ACL for Server Subnet

- Go to **Network ACLs** under **VPC**.
- Create a new ACL and associate it with the `Server` subnet.
- **Inbound Rules**:

    - Allow the following:
        - **SSH (22)**: Source `0.0.0.0/0`
        - **HTTP (80)**: Source `0.0.0.0/0`
        - **HTTPS (443)**: Source `0.0.0.0/0`
        - **ICMP (Ping)**: Source `0.0.0.0/0`
    - Block all other inbound traffic.
- **Outbound Rules**: Allow all outgoing traffic.

10. Create EC2 Instance: Web Client

- Launch a second EC2 instance:
    - **Name**: `Web Client`
    - **Subnet**: Choose `Client` subnet.
    - **Auto-assign Public IP**: Enable.
- Configure the **Security Group** with an inbound rule to allow all outbound traffic.

11. Allocate and Attach Elastic IP for Web Client

- Go to **Elastic IPs**.
- Allocate and associate a new Elastic IP with the `Web Client` instance.

12. Connect to Web Client and Install Links Browser

- Use **EC2 Instance Connect** to access the `Web Client` instance.
- Install `links`:

```
sudo apt update
sudo apt install links -y
```

- Test the connection to `Web Server` by accessing the Apache web page:
  Replace
  `<Web Server Elastic IP>` with the actual IP address.

```
links http://<Web Server Elastic IP>
```

Question 2 -

Here's a step-by-step guide to set up your subdomain, configure Route 53 for load balancing, and test the DNS setup.

1. Log in to GoDaddy

- Go to GoDaddy.
- Sign in with:
  - **Username**: `aws-ng`
  - **Password**: `Welcome1!`

2. Create a Subdomain with Registration Number

- In GoDaddy, navigate to **My Products** and go to **DNS** management.
- Select your domain and create a subdomain with your 11-digit registration number, such as `21100101001.ngaws.xyz`.

3. Create a Hosted Zone in AWS Route 53

- Open **Route 53** in the AWS Console.
- Choose **Create Hosted Zone** and configure the following:
  - **Domain Name**: Use the subdomain created, e.g., `21100101001.ngaws.xyz`.
  - **Type**: Public Hosted Zone
- After creation, Route 53 will display several **Name Servers (NS)** records. Note down any one of the NS entries.

4. Update NS Records in GoDaddy

- In GoDaddy, go to **DNS Management** for the subdomain `21100101001.ngaws.xyz`.
- Edit the NS record:

    - **Name**: `21100101001`
    - **Type**: NS
    - **Value**: Enter the Name Server copied from Route 53 (one of the NS values).
- Save changes.

5. Allocate Two Elastic IPs in AWS

- Go to **Elastic IPs** under **EC2** in the AWS Console.
- Allocate **two new Elastic IPs** and note down their IP addresses for use in the next steps.

6. Create A Records in Route 53 Hosted Zone

- Go to **Route 53** and select your hosted zone for `21100101001.ngaws.xyz`.
- Create two A records with weighted routing:
    - **Record 1**:
        - **Name**: `www`
        - **Type**: A
        - **Value**: First Elastic IP
        - **Routing Policy**: Weighted
        - **Weight**: 1
    - **Record 2**:
        - **Name**: `www`
        - **Type**: A
        - **Value**: Second Elastic IP
        - **Routing Policy**: Weighted
        - **Weight**: 1

7. Launch a New EC2 Instance (Client)

- In **EC2**, launch a new instance with the following details:
  - **Name**: `Client`
  - **Public IP**: Associate a new Elastic IP to this instance to allow internet access.
- Connect to the Client instance using **EC2 Instance Connect**.

8. Test DNS Load Balancing

- From the **Client** instance, use the `nslookup` command to test DNS load balancing:

```
nslookup www.21100101001.ngaws.xyz
```

- Run `nslookup` multiple times. With weighted routing, responses should alternate between the two Elastic IPs, showing DNS load balancing in action.

Ex - 1 - Linux commands

1. **Display information about files in the current directory**

```
ls -l
```

2. **Display the current working directory**

```
pwd
```

3. **Create a new directory**

```
mkdir <directory_name>
```

4. **Navigate between different folders**

```
cd <path_to_directory>    # Change to another directory
cd ..                     # Move up one directory level
cd                        # Go to the home directory
```

5. **Remove empty directories from the directory lists**

```
rmdir <directory_name>
```

6. **Copy files**
   - **a. Copy files from one directory to the same directory**

     ```
     cp <filename> <new_filename>
     ```

   - **b. Copy files from one directory to another directory**

     ```
     cp <filename> <path_to_target_directory>
     ```

7. **Rename and move files**
   - **a. Rename a filename to another name**

     ```
     mv <old_filename> <new_filename>
     ```

   - **b. Move a file from one directory to another**

     ```
     mv <filename> <path_to_target_directory>
     ```

8. **Delete files and directories**
   - **a. Delete individual files from a directory**

     ```
     rm <filename>
     ```

   - **b. Delete an entire directory which contains files**

```
rm -r <directory_name>
```

9. **Get basic information about the OS**

```
uname -a
```

10. **Find a file in the directory**

```
find <directory_path> -name <filename>
```

11. **Create empty files**

```
touch <filename>
```

12. **Display file contents on terminal**

```
cat <filename>           # Display full contents
head <filename>          # Display first 10 lines
tail <filename>          # Display last 10 lines
```

13. **Clear terminal**

```
clear
```

14. **Display the processes in terminal**

```
ps
```

15. **Access manual for all Linux commands**

```
man <command>
```

16. **Search for a specific string in an output**

```
grep <search_string> <filename>
```

17. **Display active processes on the terminal**

```
top
```

18. **Download files from the internet**

```
wget <URL>
```

19. **Create or update passwords for existing users**

```
passwd <username>
```

20. **View the exact location of any tool/software installed**

```
which <command_name>
```

21. **Check the details of the file system**

```
df -h
```

22. **Check lines, word count, and characters in a file using different options**

```
wc <filename>           # Show lines, words, characters
wc -l <filename>        # Show only lines
wc -w <filename>        # Show only words
wc -c <filename>        # Show only characters
```

Ex - 2 - Hadoop Installation

http://localhost:9870/

Ex - 3 - Implementing MapReduce1

Main User -

```
which sshd # /user/sbin/sshd
sudo //user/sbin/sshd # No directory - /run/sshd
sudo mkdir -p /run/sshd
ssh localhost
```

Hadoop User -

mapper.py

```
#!/usr/bin/env python3
import sys

for line in sys.stdin:
    words = line.strip().split()
    for word in words:
        print(f"{word}\t1")
```

reducer.py

```
#!/usr/bin/env python3
import sys

current_word = None
current_count = 0
word = None

for line in sys.stdin:
    word, count = line.strip().split('\t')
    count = int(count)

    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}")
```

shell commands

```
chmod +x mapper.py
chmod +x reducer.py

start-dfs.sh
start-yarn.sh
```

```
hdfs dfs -mkdir -p /BigDataLab/ex3/input # Create directory
hdfs dfs -put inputFile.txt /BigDataLab/ex3/input

ls /usr/local/hadoop/share/tools/lib/hadoop-streaming-3.3.6.jar # check if it is present

hadoop jar /usr/local/hadoop/share/hadoop/tools/lib/hadoop-streaming-3.3.6.jar
  -input /BigDataLab/ex3/input
  -output /BigDataLab/ex3/output
  -mapper /home/hd_user/LabEx/ex3/mapper.py
  -reducer /home/hd_user/LabEx/ex3/reducer.py

hdfs dfs -rm -r /BigDataLab/ex3/output # to remove output in case of errors

hdfs dfs -cat /BigDataLab/ex3/output/part-00000

stop-dfs.sh
stop-yarn.sh
```

Ex - 4(i) - Implementing MapReduce2

mapper.py

```
#!/usr/bin/env python3
import sys

for line in sys.stdin:
        parts = line.strip().split()
        date = parts[1]
        month = date[4:6]
        min_temp = parts[5]
        max_temp = parts[6]
        print(f'{month}\t{min_temp}\t{max_temp}')
```

reducer.py

```
#!/usr/bin/env python3
import sys

curr_month = None
min_temp, max_temp = float('inf'), float('-inf')

for line in sys.stdin:
        month, min_temp_str, max_temp_str = line.strip().split('\t')
        min_temp_val, max_temp_val = float(min_temp_str), float(max_temp_str)

        if curr_month == month:
                min_temp = min(min_temp, min_temp_val)
                max_temp = max(max_temp, max_temp_val)
        else:
                if curr_month is not None:
```

```
                    print(f'{curr_month}\t{min_temp}\t{max_temp}')
                curr_month = month
                min_temp = min_temp_val
                max_temp = max_temp_val

if curr_month is not None:
        print(f'{curr_month}\t{min_temp}\t{max_temp}')
```

Ex - 4(ii) - Implementing MapReduce2

mapper.py

```
#!/usr/bin/env python3
import sys
from itertools import combinations

k = int(sys.argv[1])

for line in sys.stdin:
    items = line.strip().split(',')
    items = sorted(items)
    for item in combinations(items, k):
        print(f'{",".join(item)}\t1')
```

reducer.py

```
#!/usr/bin/env python3
import sys

min_support = 2
cur_item = None
cur_count = 0

for line in sys.stdin:
    item, count = line.strip().split('\t')
    count = int(count)

    if cur_item == item:
        cur_count += count
    else:
        if cur_item and cur_count >= min_support:
            print(f'{cur_item}\t{cur_count}')
        cur_item = item
        cur_count = count

if cur_item and cur_count >= min_support:
    print(f'{cur_item}\t{cur_count}')
```

shell command

```
hadoop jar /usr/local/hadoop/share/hadoop/tools/lib/hadoop-streaming-3.3.6.jar \
  -input /BigDataLab/ex4_2/input/data.txt \
  -output /BigDataLab/ex4_2/output \
  -mapper "/home/hd_user/LabEx/ex4_2/mapper.py 2" \
  -reducer "/home/hd_user/LabEx/ex4_2/reducer.py" \
  -file /home/hd_user/LabEx/ex4_2/mapper.py \
  -file /home/hd_user/LabEx/ex4_2/reducer.py
```