Average Temperature

chaitanya@ubuntu:~$ git clone https://github.com/tomwhite/hadoop-book.git

fatal: destination path 'hadoop-book' already exists and is not an empty directory.
chaitanya@ubuntu:~$ rm -rf hadoop-book
chaitanya@ubuntu:~$ git clone https://github.com/tomwhite/hadoop-book.git

chaitanya@ubuntu:~$ cd hadoop-book/input/ncdc/all

chaitanya@ubuntu:~/hadoop-book/input/ncdc/all$ ls

chaitanya@ubuntu:~/hadoop-book/input/ncdc/all$ hdfs dfs -mkdir -p /user/ubuntu/ncdc/input

chaitanya@ubuntu:~/hadoop-book/input/ncdc/all$ hdfs dfs -put 1901.gz /ncdc/input/

put: `/ncdc/input/1901.gz': File exists
chaitanya@ubuntu:~/hadoop-book/input/ncdc/all$ hdfs dfs -rm /ncdc/input/1901.gz
Deleted /ncdc/input/1901.gz

chaitanya@ubuntu:~/hadoop-book/input/ncdc/all$ hdfs dfs -put 1901.gz /ncdc/input/

chaitanya@ubuntu:~/hadoop-book/input/ncdc/all$ hdfs dfs -rm /ncdc/input/1902.gz
Deleted /ncdc/input/1902.gz

chaitanya@ubuntu:~/hadoop-book/input/ncdc/all$ hdfs dfs -put 1902.gz /ncdc/input/

chaitanya@ubuntu:~/hadoop-book/input/ncdc/all$ nano avg_temp_mapper.py

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

for line in sys.stdin:
    if len(line) < 93:
        continue
    year = line[15:19]
    temp = line[87:92]
    quality = line[92]

    if temp != "+9999" and quality in ['0', '1', '4', '5', '9']:
        print(f"{year}\t{int(temp)}")
```

chaitanya@ubuntu:~/hadoop-book/input/ncdc/all$ nano avg_temp_reducer.py

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

current_key = None
current_vals = []

for line in sys.stdin:
    line = line.strip()
    if not line:
        continue
    key, value = line.split('\t')
    try:
        value = float(value)
    except ValueError:
        continue

    if key == current_key:
        current_vals.append(value)
    else:
        if current_key and current_vals:
            avg = sum(current_vals) / len(current_vals)
            print(f"{current_key}\t{avg:.2f}")
        current_key = key
        current_vals = [value]

if current_key and current_vals:
    avg = sum(current_vals) / len(current_vals)
    print(f"{current_key}\t{avg:.2f}")
```

chaitanya@ubuntu:~/hadoop-book/input/ncdc/all$ chmod +x avg_temp_mapper.py
avg_temp_reducer.py

chaitanya@ubuntu:~/hadoop-book/input/ncdc/all$ jps

chaitanya@ubuntu:~/hadoop-book/input/ncdc/all$ hadoop jar /usr/lib/hadoop-
mapreduce/hadoop-streaming.jar
-input /ncdc/input
-output /ncdc/output_avg_temp\
-mapper/avg_temp_napper.py \
-reducer/avg_temp_reducer.py
-file avg temp_mapper.py \
-file avg_temp_reducer.py

JAR does not exist or is not a normal file: /usr/lib/hadoop-mapreduce/hadoop-streaming.jar :

chaitanya@ubuntu:~/hadoop-book/input/ncdc/all$ find / -name "hadoop-streaming*.jar" 2>/dev/null

chaitanya@ubuntu:~/hadoop-book/input/ncdc/all$ hadoop jar /home/chaitanya/hadoop-3.4.1/share/hadoop/tools/lib/hadoop-streaming-3.4.1.jar \
-input /ncdc/input \
-output /ncdc/output_avg_temp \
-mapper avg_temp_mapper.py \
-reducer avg_temp_reducer.py \
-file avg_temp_mapper.py \
-file avg_temp_reducer.py

if Streaming Command Failed!
chaitanya@ubuntu:~/hadoop-book/input/ncdc/all$ hdfs dfs -rm -r /ncdc/output_avg_temp
Deleted /ncdc/output_avg_temp
chaitanya@ubuntu:~/hadoop-book/input/ncdc/all$ hadoop jar /home/chaitanya/hadoop-3.4.1/share/hadoop/tools/lib/hadoop-streaming-3.4.1.jar \
-input /ncdc/input \
-output /ncdc/output_avg_temp \
-mapper avg_temp_mapper.py \
-reducer avg_temp_reducer.py \
-file avg_temp_mapper.py \
-file avg_temp_reducer.py

chaitanya@ubuntu:~/hadoop-book/input/ncdc/all$ hdfs dfs -ls /ncdc/output_avg_temp

chaitanya@ubuntu:~/hadoop-book/input/ncdc/all$ hdfs dfs -cat /ncdc/output_avg_temp/part-00000

MEAN MAX TEMPERATURE

✅ New Mapper: `max_temp_per_month_mapper.py`

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

for line in sys.stdin:
    if len(line) < 93:
        continue
    year = line[15:19]
    month = line[19:21]
    temp = line[87:92]
    quality = line[92]

    if temp != "+9999" and quality in ['0', '1', '4', '5', '9']:
        key = f"{year}-{month}"
        print(f"{key}\t{int(temp)}")
```

✅ Reducer (same logic as before): `mean_temp_reducer.py`

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

current_key = None
current_vals = []

for line in sys.stdin:
    line = line.strip()
    if not line:
        continue
    key, value = line.split('\t')
    try:
        value = float(value)
    except ValueError:
        continue

    if key == current_key:
        current_vals.append(value)
    else:
        if current_key and current_vals:
            avg = sum(current_vals) / len(current_vals)
            print(f"{current_key}\t{avg:.2f}")
        current_key = key
        current_vals = [value]

if current_key and current_vals:
```

```
    avg = sum(current_vals) / len(current_vals)
    print(f"{current_key}\t{avg:.2f}")
```

🔧 Give Execution Permission

```
bash
CopyEdit
chmod +x max_temp_per_month_mapper.py mean_temp_reducer.py
```

▶ Run Hadoop Streaming Job

```
bash
CopyEdit
# First delete old output directory if exists
hdfs dfs -rm -r /ncdc/output_max_temp_per_month

# Run the job
hadoop jar /home/chaitanya/hadoop-3.4.1/share/hadoop/tools/lib/hadoop-
streaming-3.4.1.jar \
-input /ncdc/input \
-output /ncdc/output_max_temp_per_month \
-mapper max_temp_per_month_mapper.py \
-reducer mean_temp_reducer.py \
-file max_temp_per_month_mapper.py \
-file mean_temp_reducer.py
```

📄 Check Output

```
bash
CopyEdit
hdfs dfs -cat /ncdc/output_max_temp_per_month/part-00000
```

This will print the **mean max temperature per month**, formatted as:

```
python-repl
CopyEdit
1901-01 -3.42
1901-02 0.88
...
```

Let me know if you want to sort it by year or month, or filter specific years!

MEAN MAX TEMPERATURE

1. If you want mean of all minimum temperatures for every month (like the max example you gave):

You can reuse the **same mapper**, just emit the temperature values, then reducer calculates mean temperature per month.

But to find the **mean min temperature**, you probably want the **minimum temp per day first**, then average those per month.

---

2. Here is a simple version: mean of all temperature readings per month (same as max example but just for min temperatures):

If you want to find the **mean of minimum temperatures per month**, meaning you consider only temps that are less than some threshold (or all temps as they are), you can do:

Mapper: `min_temp_per_month_mapper.py`

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

for line in sys.stdin:
    if len(line) < 93:
        continue
    year = line[15:19]
    month = line[19:21]
    temp = line[87:92]
    quality = line[92]

    if temp != "+9999" and quality in ['0', '1', '4', '5', '9']:
        key = f"{year}-{month}"
        print(f"{key}\t{int(temp)}")
```

Reducer: `mean_min_temp_reducer.py`

This reducer finds the minimum temperature among all values for the month, then averages the min temperatures per day — but if only mean min temps are required directly from all temps, just calculate the mean.

If you want **mean of the minimum temperature per day per month**, then you'd need 2-step MR jobs.

For simple **mean of all min temps** (assuming temps are negative and positive), reducer calculates average:

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

current_key = None
temps = []

for line in sys.stdin:
    line = line.strip()
    if not line:
        continue
    key, value = line.split('\t')
    try:
        value = float(value)
    except ValueError:
        continue

    if key == current_key:
        temps.append(value)
    else:
        if current_key is not None and temps:
            avg = sum(temps) / len(temps)
            print(f"{current_key}\t{avg:.2f}")
        current_key = key
        temps = [value]

if current_key is not None and temps:
    avg = sum(temps) / len(temps)
    print(f"{current_key}\t{avg:.2f}")
```

Permissions and Running Job

```
bash
CopyEdit
chmod +x min_temp_per_month_mapper.py mean_min_temp_reducer.py

hdfs dfs -rm -r /ncdc/output_min_temp_per_month

hadoop jar /home/chaitanya/hadoop-3.4.1/share/hadoop/tools/lib/hadoop-
streaming-3.4.1.jar \
 -input /ncdc/input \
 -output /ncdc/output_min_temp_per_month \
 -mapper min_temp_per_month_mapper.py \
 -reducer mean_min_temp_reducer.py \
 -file min_temp_per_month_mapper.py \
 -file mean_min_temp_reducer.py
```

Output example

```
text
CopyEdit
```

```
1901-01 -5.43
1901-02 -3.12
...
```

If you want mean of daily minimum temperature per month (more accurate):

You need a two-step process like mean max example, but with min:

1. Mapper → emit `(date, temp)`
2. Reducer → find min temp per day
3. Mapper → convert day to month
4. Reducer → average daily mins per month

I can provide that if needed!

✓ 1. Create the Input File

Create a text file called `employees.txt`:

```
bash
CopyEdit
nano employees.txt
```

Paste this example data (or your actual data):

```
nginx
CopyEdit
John    HR      50000
Alice   IT      70000
Bob     HR      55000
Eve     IT      75000
Dave    IT      72000
Carol   HR      58000
```

Press `CTRL + O` then `ENTER` to save, and `CTRL + X` to exit.

✓ 2. Write the Mapper Script

Create the mapper script:

```
bash
CopyEdit
nano salary_mapper.py
```

Paste this:

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

for line in sys.stdin:
    parts = line.strip().split()
    if len(parts) != 3:
        continue
    department = parts[1]
    try:
        salary = float(parts[2])
        print(f"{department}\t{salary}")
    except ValueError:
        continue
```

Save and exit (`CTRL + O`, `ENTER`, `CTRL + X`)

✓ 3. Write the Reducer Script

```
bash
CopyEdit
nano average_salary_reducer.py
```

Paste this:

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

current_dept = None
salaries = []

for line in sys.stdin:
    line = line.strip()
    if not line:
        continue
    dept, salary = line.split('\t')
    try:
        salary = float(salary)
    except ValueError:
        continue

    if dept == current_dept:
        salaries.append(salary)
    else:
        if current_dept and salaries:
            avg = sum(salaries) / len(salaries)
            print(f"{current_dept}\t{avg:.2f}")
        current_dept = dept
        salaries = [salary]

if current_dept and salaries:
    avg = sum(salaries) / len(salaries)
    print(f"{current_dept}\t{avg:.2f}")
```

Save and exit.

✓ 4. Make the Scripts Executable

```
bash
CopyEdit
chmod +x salary_mapper.py average_salary_reducer.py
```

✓ 5. Test Locally (Optional)

```
bash
CopyEdit
cat employees.txt | ./salary_mapper.py | sort | ./average_salary_reducer.py
```

✓ 6. Upload Input File to HDFS

```
bash
CopyEdit
hdfs dfs -mkdir -p /salary/input
hdfs dfs -put -f employees.txt /salary/input/
```

✓ 7. Run the Hadoop Streaming Job

```
bash
CopyEdit
hdfs dfs -rm -r /salary/output_avg_salary

hadoop jar /home/chaitanya/hadoop-3.4.1/share/hadoop/tools/lib/hadoop-
streaming-3.4.1.jar \
-input /salary/input \
-output /salary/output_avg_salary \
-mapper salary_mapper.py \
-reducer average_salary_reducer.py \
-file salary_mapper.py \
-file average_salary_reducer.py
```

✓ 8. Display the Output

```
bash
CopyEdit
hdfs dfs -cat /salary/output_avg_salary/part-00000
```

This will print the **average salary per department**, e.g.:

```
nginx
CopyEdit
HR      54333.33
IT      72333.33
```

GENERIC MAPPER

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

# Set these two variables to the correct column indexes (0-based)
key_col = 0     # Column number for the key (e.g. year, month, department)
value_col = 1   # Column number for the value (e.g. temperature, salary)

delimiter = '\t'  # Change this to ',' for CSV, or ' ' for space-separated data

for line in sys.stdin:
    line = line.strip()
    if not line:
        continue
    parts = line.split(delimiter)

    # Basic validation: check if line has enough columns
    if len(parts) <= max(key_col, value_col):
        continue

    key = parts[key_col]
    value = parts[value_col]

    # Optional: you can add value checks here if you want to filter or convert values
    # For example, skip if value is missing or invalid
    try:
        val_float = float(value)
    except ValueError:
        continue

    print(f"{key}\t{val_float}")
```

Average-reducer

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

current_key = None
values = []

for line in sys.stdin:
    line = line.strip()
    if not line:
        continue
    try:
        key, value = line.split('\t')
        value = float(value)
    except ValueError:
        continue

    if key == current_key:
        values.append(value)
    else:
        if current_key is not None and values:
            avg = sum(values) / len(values)
            print(f"{current_key}\t{avg:.2f}")
        current_key = key
        values = [value]

if current_key is not None and values:
    avg = sum(values) / len(values)
    print(f"{current_key}\t{avg:.2f}")
```

Min – reducer

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

current_key = None
values = []

for line in sys.stdin:
    line = line.strip()
    if not line:
        continue
    try:
        key, value = line.split('\t')
        value = float(value)
    except ValueError:
        continue

    if key == current_key:
        values.append(value)
    else:
        if current_key is not None and values:
            mn = min(values)
            print(f"{current_key}\t{mn:.2f}")
        current_key = key
        values = [value]

if current_key is not None and values:
    mn = min(values)
    print(f"{current_key}\t{mn:.2f}")
```

Max-reducer

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

current_key = None
values = []

for line in sys.stdin:
    line = line.strip()
    if not line:
        continue
    try:
        key, value = line.split('\t')
        value = float(value)
    except ValueError:
        continue

    if key == current_key:
        values.append(value)
    else:
        if current_key is not None and values:
            mx = max(values)
            print(f"{current_key}\t{mx:.2f}")
        current_key = key
        values = [value]

if current_key is not None and values:
    mx = max(values)
    print(f"{current_key}\t{mx:.2f}")
```