Business problem:

Pinpointing the most popular locations through Hadoop MapReduce with focus on the year 2022.

Platform used:

Google Cloud Platform (GCP)

Dataset used:

Airbnb listings

Details of the dataset:

Shape: (87946, 75)

Columns, data types and missing values:

```
Index(['id', 'listing_url', 'scrape_id', 'last_scraped', 'source', 'name',
       'description', 'neighborhood_overview', 'picture_url', 'host_id',
       'host_url', 'host_name', 'host_since', 'host_location', 'host_about',
       'host_response_time', 'host_response_rate', 'host_acceptance_rate',
       'host_is_superhost', 'host_thumbnail_url', 'host_picture_url',
       'host_neighbourhood', 'host_listings_count',
       'host_total_listings_count', 'host_verifications',
       'host_has_profile_pic', 'host_identity_verified', 'neighbourhood',
       'neighbourhood_cleansed', 'neighbourhood_group_cleansed', 'latitude',
       'longitude', 'property_type', 'room_type', 'accommodates', 'bathrooms',
       'bathrooms_text', 'bedrooms', 'beds', 'amenities', 'price',
       'minimum_nights', 'maximum_nights', 'minimum_minimum_nights',
       'maximum_minimum_nights', 'minimum_maximum_nights',
       'maximum_maximum_nights', 'minimum_nights_avg_ntm',
       'maximum_nights_avg_ntm', 'calendar_updated', 'has_availability',
       'availability_30', 'availability_60', 'availability_90',
       'availability_365', 'calendar_last_scraped', 'number_of_reviews',
       'number_of_reviews_ltm', 'number_of_reviews_l30d', 'first_review',
       'last_review', 'review_scores_rating', 'review_scores_accuracy',
       'review_scores_cleanliness', 'review_scores_checkin',
       'review_scores_communication', 'review_scores_location',
       'review_scores_value', 'license', 'instant_bookable',
       'calculated_host_listings_count',
       'calculated_host_listings_count_entire_homes',
       'calculated_host_listings_count_private_rooms',
       'calculated_host_listings_count_shared_rooms', 'reviews_per_month'],
      dtype='object')
```

Image 1

```
id                                                int64
listing_url                                      object
scrape_id                                         int64
last_scraped                                     object
source                                           object
name                                             object
description                                      object
neighborhood_overview                            object
picture_url                                      object
host_id                                           int64
host_url                                         object
host_name                                        object
host_since                                       object
host_location                                    object
host_about                                       object
host_response_time                               object
host_response_rate                               object
host_acceptance_rate                             object
host_is_superhost                                object
host_thumbnail_url                               object
host_picture_url                                 object
host_neighbourhood                               object
host_listings_count                             float64
host_total_listings_count                       float64
host_verifications                               object
host_has_profile_pic                             object
host_identity_verified                           object
neighbourhood                                    object
neighbourhood_cleansed                           object
neighbourhood_group_cleansed                    float64
latitude                                        float64
longitude                                       float64
property_type                                    object
room_type                                        object
accommodates                                      int64
bathrooms                                       float64
bathrooms_text                                   object
bedrooms                                        float64
beds                                            float64
amenities                                        object
price                                            object
minimum_nights                                    int64
maximum_nights                                    int64
minimum_minimum_nights                          float64
maximum_minimum_nights                          float64
minimum_maximum_nights                          float64
maximum_maximum_nights                          float64
minimum_nights_avg_ntm                          float64
maximum_nights_avg_ntm                          float64
calendar_updated                                float64
has_availability                                 object
availability_30                                   int64
availability_60                                   int64
availability_90                                   int64
availability_365                                  int64
calendar_last_scraped                            object
number_of_reviews                                 int64
number_of_reviews_ltm                             int64
number_of_reviews_l30d                            int64
first_review                                     object
last_review                                      object
review_scores_rating                            float64
review_scores_accuracy                          float64
review_scores_cleanliness                       float64
review_scores_checkin                           float64
review_scores_communication                     float64
review_scores_location                          float64
review_scores_value                             float64
license                                          object
instant_bookable                                 object
calculated_host_listings_count                    int64
calculated_host_listings_count_entire_homes       int64
calculated_host_listings_count_private_rooms      int64
calculated_host_listings_count_shared_rooms       int64
reviews_per_month                               float64
dtype: object
```

Image 2

```
id                                                0
listing_url                                       0
scrape_id                                         0
last_scraped                                      0
source                                            0
name                                              0
description                                    1267
neighborhood_overview                         40756
picture_url                                       3
host_id                                           0
host_url                                          0
host_name                                         5
host_since                                        5
host_location                                 18778
host_about                                    42040
host_response_time                            28918
host_response_rate                            28918
host_acceptance_rate                          25188
host_is_superhost                               932
host_thumbnail_url                                5
host_picture_url                                  5
host_neighbourhood                            39867
host_listings_count                               5
host_total_listings_count                         5
host_verifications                                5
host_has_profile_pic                              5
host_identity_verified                            5
neighbourhood                                 40755
neighbourhood_cleansed                            0
neighbourhood_group_cleansed                  87946
latitude                                          0
longitude                                         0
property_type                                     0
room_type                                         0
accommodates                                      0
bathrooms                                     87946
bathrooms_text                                  103
bedrooms                                      32774
beds                                           1134
amenities                                         0
price                                             0
minimum_nights                                    0
maximum_nights                                    0
minimum_minimum_nights                            1
maximum_minimum_nights                            1
minimum_maximum_nights                            1
maximum_maximum_nights                            1
minimum_nights_avg_ntm                            1
maximum_nights_avg_ntm                            1
calendar_updated                              87946
has_availability                                  0
availability_30                                   0
availability_60                                   0
availability_90                                   0
availability_365                                  0
calendar_last_scraped                             0
number_of_reviews                                 0
number_of_reviews_ltm                             0
number_of_reviews_l30d                            0
first_review                                  22158
last_review                                   22158
review_scores_rating                          22158
review_scores_accuracy                        23093
review_scores_cleanliness                     23081
review_scores_checkin                         23125
review_scores_communication                   23095
review_scores_location                        23125
review_scores_value                           23126
license                                       87945
instant_bookable                                  0
calculated_host_listings_count                    0
calculated_host_listings_count_entire_homes       0
calculated_host_listings_count_private_rooms      0
calculated_host_listings_count_shared_rooms       0
reviews_per_month                             22158
dates_2022                                        0
```

Image 3

Data cleaning:

The data cleaning was done on python.

1. Since the focus was on the year 2022, the columns 'first_review' and 'last_review' were considered.
    - They were first converted to datetime format.
    - A new column dates_2022 was created which combined the dates mentioned in these two columns.
    - The cell was mentioned as ‘Null’ if the date did not fall in the year 2022.
    - Sample output column looks as the one in image 4.
2. A sub dataset was made by using the columns 'neighbourhood_cleansed', 'number_of_reviews','dates_2022'
    - Cells having the values as ‘Null’ in the column ‘dates_2022’ were dropped as they did not belong to year 2022. There were total 72157 such values.
    - There were no null values in the 'number_of_reviews' column.
    - The final dataset was saved as csv file by the name of ‘Review_counts_2022_final.csv’.
    - The shape of this file was (15789, 3)
    - The code snippet of this step is shown in image 5.

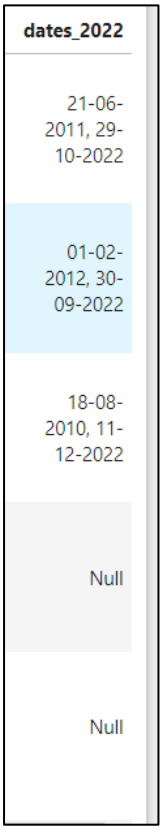

Image 4

```
#Number of reviews
df_reviews_counts = df[['neighbourhood_cleansed', 'number_of_reviews','dates_2022']]
df_reviews_counts.shape
```

```
(87946, 3)
```

```
df_reviews_counts.isnull().sum()
```

```
neighbourhood_cleansed    0
number_of_reviews         0
dates_2022                0
dtype: int64
```

```
#Dropping the rows with 'Null' as the value in dates_2022 column
df_reviews_counts_dropnull = df_reviews_counts[df_reviews_counts['dates_2022'] != 'Null']
df_reviews_counts_dropnull.shape
```

```
(15789, 3)
```

```
df_reviews_counts_dropnull.to_csv('Review_counts_2022_final.csv')
```

Image 5

Approach to solution:

- The popular locations were aimed to be found on the basis of number of reviews in the year 2022.

Details of map reduce:

- Step 1: Input split
  - Each row of the dataset would be considered as an input consisting of location name and count of reviews.
- Step 2: Mapping
  - Each location is mapped to its count of reviews.
- Step 3: Shuffling
  - To sort the results of mapping in alphabetical order
- Step 4: Reducer
  - The output of step 3 was fed into the reducer.
  - For every location, the count of reviews is maintained and top 5 locations with maximum number of reviews is displayed as the output.
- The code snippet for Step 1 and 2 can be seen in image 6 Mapper.py
- To execute step 3, we used simple 'sort' command.
- The code snippet for Step 4 can be seen in image 7 Reducer_for_counts.py

```
#!/usr/bin/env python3
"""mapper.py"""


import sys
import csv

# Input comes from standard input (stdin)
for line in sys.stdin:
    # Create a CSV reader object
    reader = csv.reader([line])
    # Read the row
    for row in reader:
        # Extract location name and review score
        location = row[0]
        score = float(row[1])  # Assuming the score is the second column
        # Emit the location as the key and the score as the value
        print('%s\t%s' % (location, score))
```

Image 6: Mapper.py

```python
#!/usr/bin/env python
"""reducer.py"""

import sys
from collections import defaultdict
from heapq import nlargest

# Dictionary to store the total count for each location
location_counts = defaultdict(float)

# Input comes from standard input (stdin)
for line in sys.stdin:
    # Remove leading and trailing whitespace
    line = line.strip()
    # Parse the input we got from mapper.py
    location, count = line.split('\t', 1)
    # Update the total count for the location
    location_counts[location] += float(count)

# Find the top 5 locations with the maximum number of counts
top_5_locations = nlargest(5, location_counts, key=location_counts.get)

# Print the top 5 locations with the maximum number of counts
print("Top 5 Locations with Maximum Number of Counts:")
for loc in top_5_locations:
    print(f"Location: {loc}, Number of Counts: {location_counts[loc]}")
```

Image 7: Reducer_for_counts.py

Mapper.py explained

This code in image 6 is a Python script typically used as a mapper function in a MapReduce job. It reads input from standard input (stdin) and emits key-value pairs to standard output (stdout).

1. `#!/usr/bin/env python3`: This is a shebang line, specifying the interpreter to be used to execute the script. In this case, it indicates that the script should be executed using the Python 3 interpreter. When the script is run from the command line, the operating system will use the interpreter specified here to execute the script.

2. `"""mapper.py"""`: This is a docstring, providing a brief description of the script. It's enclosed within triple quotes and doesn't affect the functionality of the code but serves as documentation.

3. `import sys`: This imports the sys module, which provides access to some variables used or maintained by the Python interpreter and to functions that interact strongly with the interpreter.

4. `import csv`: This imports the csv module, which provides functions for reading and writing CSV files.

5. The script then iterates over each line of input received from standard input (stdin) using a for loop: `for line in sys.stdin:`.

6. Inside the loop, it creates a CSV reader object `reader` using `csv.reader([line])`. This is done to parse the input line as a CSV row, assuming that the input lines are comma-separated values.

7. It then iterates over each row in the CSV reader object `for row in reader:`. In this case, since we're reading one line at a time from stdin, there will typically only be one row.

8. Inside the inner loop, it extracts the location name and review score from the row. It assumes that the location is in the first column (`location = row[0]`) and the score is in the second column (`score = float(row[1])`).

9. Finally, it emits the location as the key and the score as the value, separated by a tab character (`\t`), using `print('%s\t%s' % (location, score))`. This output is what will be passed to the reducer function in a MapReduce job.

Reducer_for_counts.py explained

This code as shown in image 7 is a Python script designed to process input data and find the top 5 locations with the highest counts. Here's a breakdown of how it works:

1. `import sys`: This imports the sys module, which provides access to some variables used or maintained by the Python interpreter and to functions that interact with the interpreter.

2. `from collections import defaultdict`: This imports the defaultdict class from the collections module. defaultdict is a subclass of the built-in dict class. It overrides one method and adds one writable instance variable. The defaultdict will create a new entry for a key if it doesn't exist, initializing it to a default value.

3. `from heapq import nlargest`: This imports the nlargest function from the heapq module. `nlargest` is used to find the n largest elements from a collection.

4. `location_counts = defaultdict(float)`: This creates a defaultdict object named `location_counts` where the default value for each key (location) is set to 0.0 (float).

5. The script then starts reading input from the standard input (stdin) line by line using a for loop: `for line in sys.stdin:`. Each line is stripped of leading and trailing whitespace.

6. `location, count = line.split('\t', 1)`: This splits each line into two parts based on the tab character ('\t'). The first part is assigned to `location`, and the second part is assigned to `count`. The `1` argument in `split()` limits the split to one occurrence of '\t', ensuring that even if the count contains tabs, they won't affect the splitting.

7. `location_counts[location] += float(count)`: This updates the count for each location by adding the count from the current line. `float(count)` is used to convert the count from string to float before adding it to the existing count for that location.

8. After processing all input lines, `nlargest(5, location_counts, key=location_counts.get)` is used to find the top 5 locations with the highest counts. It takes three arguments: the number of

elements to retrieve (5), the collection to search (location_counts), and a key function (location_counts.get) which is used to extract a comparison key from each element.

9. Finally, the script prints out the top 5 locations along with their counts using a loop.

Execution phase:

- The clusters using 'Dataproc' were activated and SSH was established.

File upload

- The dataset, mapper.py and reducer_for_counts.py were uploaded using simple 'upload file' option as shown in image 8.
- The command 'ls' confirms that the files have been uploaded successfully. This can be seen in image 9.

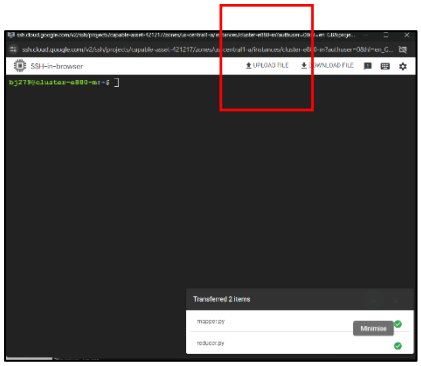

Image 8

```
bj279@cluster-e880-m:~$ ls
Review_counts_2022_final.csv  Review_rating_2022_final.csv  mapper.py  reducer.py  reducer_for_counts.py
```

Image 9

mapper.py and output

- For mapper.py, the contents for the dateset 'Review_counts_2022_final.csv' were fed as input. For this we used 'cat' command and a '|' (pipe) symbol to make the pipeline.
- The final command can be seen in image 10.
- The counts of reviews are mapped to their respective locations. This can be seen in image 11.

```
bj279@cluster-e880-m:~$ cat Review_counts_2022_final.csv | python mapper.py
```

Image 10

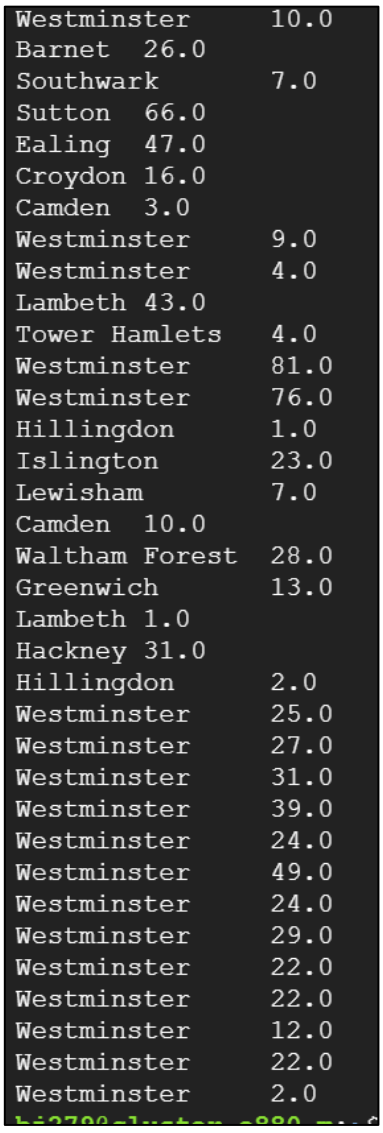

Image 11

Shuffling/Sorting and output

- The output of the mapper.py was fed as input to 'sort' and the result was sorted on the basis of alphabetical order of the locations.
- The command and output can be seen in images 12 and 13 respectively.

Image 12

```
Westminster     9.0
Westminster     9.0
Westminster     9.0
Westminster     9.0
Westminster     9.0
Westminster     9.0
Westminster     9.0
Westminster     9.0
Westminster     9.0
Westminster     9.0
Westminster     9.0
Westminster     9.0
Westminster     9.0
Westminster     9.0
Westminster     9.0
Westminster     9.0
Westminster     9.0
Westminster     9.0
Westminster     9.0
Westminster     9.0
Westminster     90.0
Westminster     90.0
Westminster     90.0
Westminster     91.0
Westminster     91.0
Westminster     92.0
Westminster     94.0
Westminster     95.0
Westminster     96.0
Westminster     97.0
Westminster     98.0
Westminster     99.0
Westminster     99.0
Westminster     99.0
Westminster     99.0
bj279@cluster-e880-m:~$
```

Image 13

Reducer_for_counts.py and output

- The output of 'shuffling' was fed as input to the 'reducer_for_counts.py' as shown in image 14.
- The output of reducer_for_counts.py can be seen in image 15.

```
bj279@cluster-e880-m:~$ cat Review_counts_2022_final.csv | python mapper.py | sort | python reducer_for_counts.py
```

Image 14

```
bj279@cluster-e880-m:~$ cat Review_counts_2022_final.csv | python mapper.py | sort | python reducer_for_counts.py
Top 5 Locations with Maximum Number of Counts:
Location: Westminster, Number of Counts: 39324.0
Location: Camden, Number of Counts: 23788.0
Location: Tower Hamlets, Number of Counts: 22827.0
Location: Kensington and Chelsea, Number of Counts: 19088.0
Location: Southwark, Number of Counts: 16905.0
bj279@cluster-e880-m:~$
```

Image 15

Conclusion:

Based on the number of reviews in the year 2022, following are the top 5 popular locations:

1. Westminster
2. Camden
3. Tower of Hamlets
4. Kensington and Chelsea
5. Southwark

Unresolved bottlenecks:

We encountered the following errors during the execution of the map reduce on the HDFS. These issues were unresolved at the time of submission.

Case 1:

Unable to read the files.

```
bj279@cluster-e880-m:~$ hdfs dfs -cat Review_rating_2022_final.csv
cat: /user/bj279 (is not a directory)
bj279@cluster-e880-m:~$ hdfs dfs -cat /data/Review_rating_2022_final.csv | /data/mapper.py
-bash: /data/mapper.py: No such file or directory
cat: Unable to write to output stream.
bj279@cluster-e880-m:~$ hdfs dfs -cat /data/Review_rating_2022_final.csv | python /data/mapper.py
python: can't open file '/data/mapper.py': [Errno 2] No such file or directory
cat: Unable to write to output stream.
bj279@cluster-e880-m:~$ ^C
bj279@cluster-e880-m:~$ hdfs dfs -cat /data/Review_rating_2022_final.csv | python /data/mapper.py
python: can't open file '/data/mapper.py': [Errno 2] No such file or directory
hdfs cat: Unable to write to output stream.
bj279@cluster-e880-m:~$ hdfs dfs -ls /data/
Found 3 items
-rw-r--r--   2 bj279 hadoop     258831 2024-05-11 17:31 /data/Review_rating_2022_final.csv
-rw-r--r--   2 bj279 hadoop        514 2024-05-11 17:39 /data/mapper.py
-rw-r--r--   2 bj279 hadoop       1210 2024-05-11 17:39 /data/reducer.py
```

```
bj279@cluster-e880-m:~$ hdfs dfs -cat /data/Review_rating_2022_final.csv | python3 /data/mapper.py
python3: can't open file '/data/mapper.py': [Errno 2] No such file or directory
cat: Unable to write to output stream.
```

Case 2: Ref (Lab document)

Stream command failed

```
bj279@cluster-e880-m:~$ find /usr/ -name 'hadoop-streaming*.jar' 2>/dev/null
/usr/lib/hadoop/hadoop-streaming.jar
/usr/lib/hadoop/hadoop-streaming-3.3.6.jar
bj279@cluster-e880-m:~$
```

```
bj279@cluster-e880-m:~$ hadoop jar /usr/lib/hadoop/hadoop-streaming-3.3.6.jar -file /data/mapper.py -mapper mapper.py -file /data/reducer.py -reducer reduce
r.py -input /data/Review_rating_2022_final.csv -output /data/Output_review_rating
2024-05-11 20:07:51,513 WARN streaming.StreamJob: -file option is deprecated, please use generic option -files instead.
File: file:/data/mapper.py does not exist.
Try -help for more information
Streaming Command Failed!
```

Changed the code using the below reference but the error persisted.

The `-file` option from hadoop-streaming only works for local files. Note however, that its help text mentions that the `-file` flag is deprecated in favor of the generic `-files` option. Using the generic `-files` option allows us to specify a remote (hdfs / gs) file to stage. Note also that generic options must precede application specific flags.

Your invocation would become:

```
hadoop jar /usr/lib/hadoop-mapreduce/hadoop-streaming.jar \
    -files gs://bucket-name/intro_to_mapreduce/mapper_prod_cat.py,gs://bucket-name/ir
    -mapper mapper_prod_cat.py \
    -reducer reducer_prod_cat.py \
    -input gs://bucket-name/intro_to_mapreduce/purchases.txt \
    -output gs://bucket-name/intro_to_mapreduce/output_prod_cat
```

Ref: https://stackoverflow.com/questions/48003377/error-when-running-python-map-reduce-job-using-hadoop-streaming-in-google-cloud

```
bj279@cluster-e880-m:~$ hadoop jar /usr/lib/hadoop/hadoop-streaming-3.3.6.jar \
    -files /data/mapper.py,/data/reducer.py \
    -mapper mapper.py \
    -reducer reducer.py \
    -input /data/Review_rating_2022_final.csv \
    -output /data/Output_review_rating.txt
Exception in thread "main" java.io.FileNotFoundException: File /data/mapper.py does not exist
        at org.apache.hadoop.fs.RawLocalFileSystem.deprecatedGetFileStatus(RawLocalFileSystem.java:915)
        at org.apache.hadoop.fs.RawLocalFileSystem.getFileLinkStatusInternal(RawLocalFileSystem.java:1236)
        at org.apache.hadoop.fs.RawLocalFileSystem.getFileStatus(RawLocalFileSystem.java:905)
        at org.apache.hadoop.fs.FilterFileSystem.getFileStatus(FilterFileSystem.java:462)
        at org.apache.hadoop.util.GenericOptionsParser.validateFiles(GenericOptionsParser.java:463)
        at org.apache.hadoop.util.GenericOptionsParser.validateFiles(GenericOptionsParser.java:408)
        at org.apache.hadoop.util.GenericOptionsParser.processGeneralOptions(GenericOptionsParser.java:340)
        at org.apache.hadoop.util.GenericOptionsParser.parseGeneralOptions(GenericOptionsParser.java:581)
        at org.apache.hadoop.util.GenericOptionsParser.<init>(GenericOptionsParser.java:182)
        at org.apache.hadoop.util.GenericOptionsParser.<init>(GenericOptionsParser.java:164)
        at org.apache.hadoop.util.ToolRunner.run(ToolRunner.java:76)
        at org.apache.hadoop.util.ToolRunner.run(ToolRunner.java:97)
        at org.apache.hadoop.streaming.HadoopStreaming.main(HadoopStreaming.java:50)
        at java.base/jdk.internal.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
        at java.base/jdk.internal.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)
        at java.base/jdk.internal.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
        at java.base/java.lang.reflect.Method.invoke(Method.java:566)
        at org.apache.hadoop.util.RunJar.run(RunJar.java:328)
        at org.apache.hadoop.util.RunJar.main(RunJar.java:241)
```

Referring to the below articles, it was mentioned that the mapper and reducer donot need to be uploaded on the hdfs.

> If you checked the instructions carefully on the link,
>
> ```
> hduser@ubuntu:/usr/local/hadoop$ bin/hadoop jar contrib/streaming/hadoop-*streaming*
> ```
>
> there it clearly shows there is no need to copy the mapper.py and reducer.py to the HDFS, you can link both the files from the local filesystem: as /path/to/mapper. I am sure you can avoid the above error.

https://www.michael-noll.com/tutorials/writing-an-hadoop-mapreduce-program-in-python/

https://stackoverflow.com/questions/15353252/running-the-python-code-on-hadoop-failed

The error persisted.

```
bj279@cluster-e880-m:~$ hadoop jar /usr/lib/hadoop/hadoop-streaming-3.3.6.jar \
    -files /mapper.py,/reducer.py \
    -mapper mapper.py \
    -reducer reducer.py \
    -input /data/Review_rating_2022_final.csv \
    -output /data/Output_review_rating
Exception in thread "main" java.io.FileNotFoundException: File /mapper.py does not exist
        at org.apache.hadoop.fs.RawLocalFileSystem.deprecatedGetFileStatus(RawLocalFileSystem.java:915)
        at org.apache.hadoop.fs.RawLocalFileSystem.getFileLinkStatusInternal(RawLocalFileSystem.java:1236)
        at org.apache.hadoop.fs.RawLocalFileSystem.getFileStatus(RawLocalFileSystem.java:905)
        at org.apache.hadoop.fs.FilterFileSystem.getFileStatus(FilterFileSystem.java:462)
        at org.apache.hadoop.util.GenericOptionsParser.validateFiles(GenericOptionsParser.java:463)
        at org.apache.hadoop.util.GenericOptionsParser.validateFiles(GenericOptionsParser.java:408)
        at org.apache.hadoop.util.GenericOptionsParser.processGeneralOptions(GenericOptionsParser.java:340)
        at org.apache.hadoop.util.GenericOptionsParser.parseGeneralOptions(GenericOptionsParser.java:581)
        at org.apache.hadoop.util.GenericOptionsParser.<init>(GenericOptionsParser.java:182)
        at org.apache.hadoop.util.GenericOptionsParser.<init>(GenericOptionsParser.java:164)
        at org.apache.hadoop.util.ToolRunner.run(ToolRunner.java:76)
        at org.apache.hadoop.util.ToolRunner.run(ToolRunner.java:97)
        at org.apache.hadoop.streaming.HadoopStreaming.main(HadoopStreaming.java:50)
        at java.base/jdk.internal.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
        at java.base/jdk.internal.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)
        at java.base/jdk.internal.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
        at java.base/java.lang.reflect.Method.invoke(Method.java:566)
        at org.apache.hadoop.util.RunJar.run(RunJar.java:328)
        at org.apache.hadoop.util.RunJar.main(RunJar.java:241)
```