**Python:**

The file name is **Data\_Cleaning\_code.py**. The codes are available as chunks and can be executed using a jupyter notebook.

1. Data Cleaning
2. Removing Outliers
3. Cross Join Verification
4. Pearson Correlation Graph

**Java:**

Assuming the below 3 jars are available in the local folder. Please execute the jar in the same order mentioned in the document to get appropriate results.

1. **ReduceJoin.jar:**

It contains 2 mapper files, 1 reducer and 1 main java file.

**hadoop jar ReduceJoin.jar ReduceJoin /input\_file\_1 /input\_file\_2 /output\_dir\_1**

1. **PearsonJob.jar:**

It contains 1 reducer, 1 mapper, 3 dependencies and 1 main java file. The output from first .jar file is passed as input (1st argument) for this jar.

**hadoop jar PearsonJob.jar PearsonJob /output\_dir\_1 /output\_dir\_2**

1. **GroupMR.jar:**

It contains a mapper, reducer and main class in a single java file. The output from first .jar file is passed as input (1st argument) for this jar.

**hadoop jar GroupMR.jar GroupMR /output\_dir\_1 /output\_dir\_3**

Suppose, if you face java heap space error while executing the code. Take a sample 100000 records from the file of output\_dir\_1 and run the code.

Set Java heap memory to **2gb** if you are using the OpenStack **m1.medium** machine.

**export \_JAVA\_OPTIONS="-Xmx2g"**

To take sample records and rerun the jar execute the below commands:

**hadoop fs -get output\_dir\_1/part-r-00000 .**

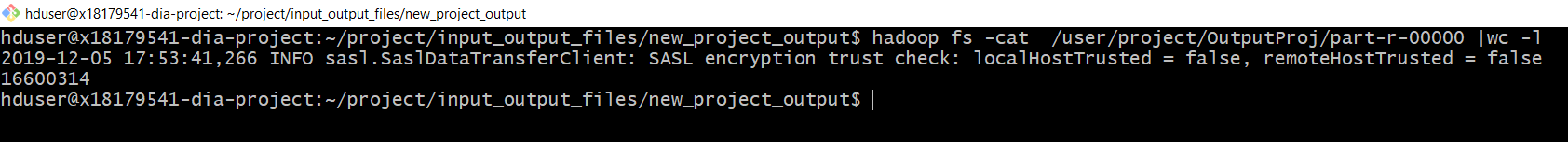
**head -n100000 part-r-00000 > test\_data\_100000**

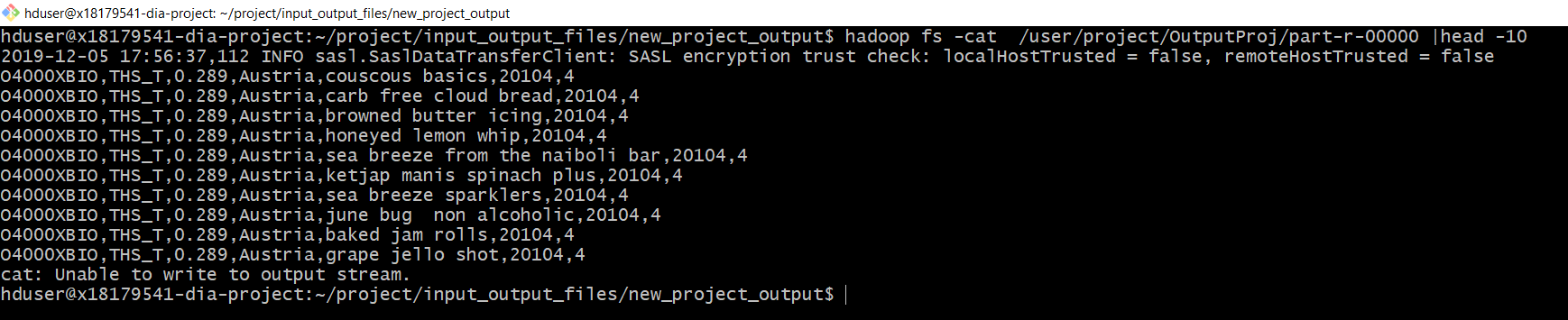
**hadoop fs -copyFromLocal test\_data\_100000 /output\_dir\_1**

**hadoop jar GroupMR.jar GroupMR /output\_dir\_1/ test\_data\_100000 /output\_dir\_3**

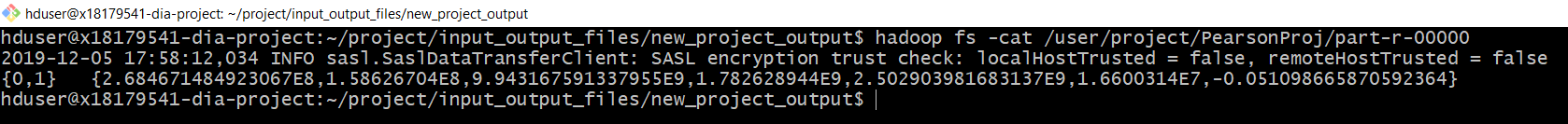
**ScreenShots of the output:**

1. Word count and sample data of the Joined dataset.





1. Pearson Correlation Output.



1. Group by function word count and sample output data for randomly chosen 100000 records as input from 1st Jar.

