1.)

a.) For this query: Mapper has a key of First + Last and values of EID, AID and Age
Reducer will receive the First + Last key and the values EID, AID and Age.

b.) For this query: Mapper 1-a has key lo_orderdate the values are lo_extendedprice, d_yearmonth and lo_discount. Mapper 1-b has key d_datekey the value will be lo_extendedprice. Reducer receives sum of lo_extendedprice for each lo_orderdate when the d_yearmonth is 'Feb1996' and lo_discount has a value of 5.

c.) For this query: Mapper has key d_month and the value is d_year. Reducer receives each d_month and groups each d_month and is ordered by the count of d_year.

2.)

a.) For 1 worker node: 79 (blocks) * 1 (minute) * 60 (seconds) + 5000 * 1 (second) =
9740 seconds or 151.23 minutes or 2.52 hours.

b.) For 30 worker nodes: 9740(sec) / 30 =
324.6 seconds or 5.411 minutes

c.) For 50 worker nodes: 9740(sec) / 50 =
194.8 seconds or 3.246 minutes

d.) For 100 worker nodes: 9740(sec) / 100 =
97.4 seconds or 1.623 minutes

e.) Yes, if the replication factor is changed, then there might be an increase of time for the reducer tasks. However, there may be an improvement in overall performance.

3.)

a.i.) In this situation, the NameNode will redistribute the blocks that are on the failed node to the remaining 7 nodes and may replicate each of the blocks again due to the replication factor of 3.

a.ii.) For this case, the failed job, either a failed mapper or failed reducer, will be sent to another node and to be started again.

b.) The mapper stores output key-value pairs on the local disk / local file system, before sending this to the reducers.

c.) No, all of the data should be collected from the mapper before reduce starts. There are tasks such as sort which cannot be completed until the mapper has completed. There is a

task named “shuffle” which can be completed while the mapper is working, because shuffle only “transfers the data”.

4.)

a.) p = 3, q = 17
n = 3 * 17 = 51
Ø(n) = (3 – 1) * (17- 1)
= 32

gcd(e, 32) = 1 and e<32
e = 23
d = 29
Public Key - KU = {e,n}
= {23, 51}

Private Key - KR = {d,n}
= {29, 51}

b.) M = 49
ciphertext C = 49^{23} mod 51
ciphertext C = 25

c.) M = 25^{19} mod 51
M = 19

d.) If the encrypted message is larger than n, then the remainder may truncate the value. This means that after decryption, the initial message will never be returned. If the value is truncated, that could give a host of potential messages. To resolve this issue, large messages must be cut into smaller blocks.

5.)

part_transform.py:

```
#!/usr/bin/python
import sys

for line in sys.stdin:
    #line = line.replace(' ','~').replace('#','~')
    line = line.strip().split('\t')
    seven = line[6].split(' ')
    line[6] = ','.join([seven[2], seven[1], seven[0]])

    print '~'.join([line[0].replace(' ','~'), line[1].replace(' ','~'), line[2].replace('#','~'),
        line[3].replace('#','~$
```

```
create table part (
  p_partkey     int,
  p_name        varchar(22),
  p_mfgr        varchar(6),
  p_category    varchar(7),
  p_brand1      varchar(9),
  p_color       varchar(11),
  p_type        varchar(25),
  p_size        int,
  p_container   varchar(10)
)
ROW FORMAT DELIMITED FIELDS
TERMINATED BY'|' stored as textfile;
```

```
load data local inpath '/home/ec2-user/part.tbl' overwrite into table part;
```

```
add file /home/ec2-user/part_transform.py;
```

```
select transform (p_partkey, p_name, p_mfgr, p_category, p_brand1, p_color, p_type, p_size,
        p_container) using 'part_transform.py' as (p_partkey, p_name, p_mfgr, p_category,
        p_brand1, p_color, p_type, p_size, p_container) from part;
```

```
create table partswapped (
```

```
  p_partkey     int,
  p_name        varchar(22),
  p_mfgr        varchar(6),
  p_category    varchar(7),
  p_brand1      varchar(9),
  p_color       varchar(11),
  p_type        varchar(25),
  p_size        int,
  p_container   varchar(10)
)
ROW FORMAT DELIMITED FIELDS
TERMINATED BY'\t' stored as textfile;
```

```
insert overwrite table partswapped select transform (p_partkey, p_name, p_mfgr, p_category,
        p_brand1, p_color, p_type, p_size, p_container) using 'part_transform.py' as (p_partkey,
        p_name, p_mfgr, p_category, p_brand1, p_color, p_type, p_size, p_container) from part;
```

```
hive> insert overwrite table partswapped select transform (p_partkey, p_name, p_mfgr, p_category, p_brand1, p_color, p_type, p_size, p_containe
r) using 'part_transform.py' as (p_partkey, p_name, p_mfgr, p_category, p_brand1, p_color, p_type, p_size, p_container) from part;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i.e.
spark, tez) or using Hive 1.X releases.
Query ID = ec2-user_20220429013447_c3b91315-c04c-46c0-ab9c-47a5951ceef6
Total jobs = 3
Launching Job 1 out of 3
Number of reduce tasks is set to 0 since there's no reduce operator
Starting Job = job_1651176821173_0009, Tracking URL = http://ip-172-31-30-225.us-east-2.compute.internal:8088/proxy/application_1651176821173_0
009/
Kill Command = /home/ec2-user/hadoop-2.6.4/bin/hadoop job  -kill job_1651176821173_0009
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 0
2022-04-29 01:34:55,690 Stage-1 map = 0%,  reduce = 0%
2022-04-29 01:35:09,805 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 7.48 sec
MapReduce Total cumulative CPU time: 7 seconds 480 msec
Ended Job = job_1651176821173_0009
Stage-4 is selected by condition resolver.
Stage-3 is filtered out by condition resolver.
Stage-5 is filtered out by condition resolver.
Moving data to: hdfs://localhost/user/hive/warehouse/partswapped/.hive-staging_hive_2022-04-29_01-34-47_762_8618782287860255224-1/-ext-10000
Loading data to table default.partswapped
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1   Cumulative CPU: 7.48 sec   HDFS Read: 17146867 HDFS Write: 5400084 SUCCESS
Total MapReduce CPU Time Spent: 7 seconds 480 msec
OK
Time taken: 23.428 seconds
hive>
```

6.) Rows after creating table and DUMP Count:

```
process : 1
(34174)
```

Size of the newly created file. I opted for “ThreeColExtract2”, as I wanted to rename the original and save it, but I’m unsure how to rename files in Hadoop:

```
drwxr-xr-x   - ec2-user supergroup          0 2022-04-29 02:34 ThreeColExtract2
-rw-r--r--   1 ec2-user supergroup          0 2022-04-29 02:34 ThreeColExtract2/_SUCCESS
-rw-r--r--   1 ec2-user supergroup     627867 2022-04-29 02:34 ThreeColExtract2/part-m-00000
```